JHONATAN GEREMIAS

DETECÇÃO EM TEMPO REAL DE PORNOGRAFIA EM VÍDEOS USANDO CNN E ANÁLISE DE CONTEXTO BASEADA EM JANELAS DE FRAMES

CURITIBA

2020

JHONATAN GEREMIAS

DETECÇÃO EM TEMPO REAL DE PORNOGRAFIA EM VÍDEOS USANDO CNN E ANÁLISE DE CONTEXTO BASEADA EM JANELAS DE FRAMES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Pontifícia Universidade Católica do Paraná

Programa de Pós-Graduação em Informática

Orientador: Alceu de Souza Britto Junior

Coorientador: Altair Olivo Santin

CURITIBA

2020

Agradecimentos

Primeiramente agradeço a Deus que me concedeu a vida dando forças e condições para chegar até aqui, sou grato por me conduzir nesta jornada permitindo superar todos os obstáculos e conseguir ir além dos meus limites. Obrigado pelos dias bons e por me confortar nos momentos difíceis.

A minha família por todo apoio incondicional, amor, paciência e compreensão, o suporte necessário para vencer mais esta etapa. Um agradecimento especial aos meus pais João Gilberto Geremias e Iolanda Vicentina Xavier Geremias que me educaram e conduziram pelos caminhos que me trouxeram até aqui, incentivaram nos estudos, são os grandes responsáveis por minha existência e pela pessoa que me tornei. A meus queridos filhos Luan Eduardo Lemos Geremias e Leonardo Lemos Geremias por todo amor e imensa felicidade que me proporcionam em todos os momentos, minha grande motivação em buscar a ser uma melhor versão de mim. Aos meus irmãos Kelly Cristina Xavier Geremias e João Gilberto Geremias Junior pela parceria, conversas e conselhos.

Agradeço imensamente aos professores Dr. Alceu de Souza Britto Junior e Dr. Altair Olivo Santin a qual tenho muita admiração e prestígio. Tive o privilégio e a oportunidade de trabalhar sob a orientação de ambos os professores. Sou eternamente grato ao tempo e atenção que foi a mim dedicada. Foram inestimáveis contribuições, inúmeras ideias, sugestões, comentários, esclarecimentos, críticas construtivas e discussões, sem as quais não seria possível o desenvolvimento deste trabalho. Sobretudo agradeço a paciência e confiança em mim depositada. Agradeço também ao professor Dr. Eduardo Kugler Viegas, com sabedoria em suas considerações e conselhos enriqueceram meu trabalho. Aos professores Dr. Luiz Eduardo Soares de Oliveira e Dr. Paulo Rodrigo Cavalin que fizeram considerações importantes na banca o que permitiu aperfeiçoar o meu trabalho.

Agradeço a NVIDIA por disponibilizar a GPU que viabilizou a realização dos experimentos, em especial ao Flávio de Almeida e Silva por compartilhar os recursos para que isso fosse possível. Entre os diversos projetos que fornecem incentivo a pesquisa, gratidão ao Grupo Marista, CNPq, Fundação Araucária, FINEP e CAPES.

Também um agradecimento especial: Ao Cleverton Juliano Alves Vicentini, Roberta Rafaela Sotero Costa e Zacarias Curi Filho pela revisão realizada no documento, o texto agora está fluindo melhor, recebeu um polimento. A Cheila Cristina e Flávia

Beuting por todo auxílio e parceria na secretaria. Aos meus companheiros de trabalho e amigos no suporte Roger Robson, Nicolas de Paula e Anderson Bertling. A todos os professores do PPGia que transmitiram seu conhecimento.

Agradeço aos amigos do PPGIA: Allan da Silva Espindola, André Brun, Andréia Marini, Bruno de Barros Bullé, Bruno Souza, Cleber Olivo Santin, Cheila Cristina, Cleverton Vicentini, Denise Sato, Diogo Olsen, Edenilson José da Silva, Eduardo Viegas, Elias Carvalho, Elio Ribeiro Faria, Erich Malinowski, Estefânia Fuzyi, Evelyn Henkel, Flávia Beuting, Flávio de Almeida e Silva, Gregory Moro Puppi Wanderley, Gustavo Bonacina, Heitor Gomes, Irapuru Florido e Débora, Jackson Mallmann, Jean-Paul Barddal, João Pedro Rodrigues, José Eduardo Nunes Lino, Jurandir dos Santos, Karine Nacano, Kelly Wiggers, Lilian Ferreira, Lucas Martiniano, Luiz Giovanini, Marcelo Pereira, Marcia Pascutti, Mariza Dosciatti, Marcos Monteiro, Milena Alves, Nicolas de Paula, Patrícia Antoniolli, Priscila Louise Leysler Santin, Rafael Cruz Ribeiro, Rafael Faria de Azevedo, Rodolfo Botto, Rodolfo Pereira, Rodrigo Siega, Rosana Lachowski, Sandoval Ruppel, Sediane Hernandes, Sidnei Schuindt, Vilmar Abreu, Viviane Dal Molin, Voncarlos Araújo e Zacarias Curi.

E outras pessoas importantes, amigos que estiveram sempre de portas abertas e dispostos a me ajudar, Adilson Galiano, Aline Velozo, Andressa Zambiassi, Cleriston Simette, Daniel Jansen, Edson Kageyama, Everton Esteves, Jocemir Bornholdt, Luiz Henrique, Marcelo de Jesus, Nathália Giovana e Silvio Viezzer.

Por fim, seria injusto se esquece alguém, então agradeço a todos que de forma direta ou indiretamente contribuíram para que este trabalho fosse possível. Muito Obrigado!!!

Resumo

Detecção automatizada de vídeos pornográficos tem sido objeto de vários estudos na literatura. Geralmente, essa mídia é dividida em *frames*, que posteriormente são classificados separadamente. Embora tais abordagens sejam amplamente estudadas, não são capazes de identificar conteúdo sensível dependendo do contexto. Por exemplo, de acordo com o contexto, uma pessoa despida pode estar tomando banho ou participando de uma cena em um contexto sexual. Assim, este trabalho propõe uma abordagem computacionalmente leve de detecção de conteúdos sensíveis em vídeos. Na proposta, primeiramente utiliza-se uma CNN (*Convolutional Neural Network*) treinada para detecção de imagens pornográficas e classificação de *frames* individuais dos vídeos. Posteriormente, um algoritmo de reconhecimento de padrões é alimentado com a saída da CNN, extraindo informações relacionadas com o contexto, em uma abordagem baseada em janela de tempo. O método proposto é avaliado em um novo *dataset*, composto por mais de 476 mil *frames* classificados individualmente, construídos a partir de mais de 14 mil vídeos. As avaliações mostram que as CNNs tradicionais, treinadas em imagens, não são capazes de detectar vídeos pornográficos de acordo com o seu contexto. Além disso, a abordagem proposta melhora a acurácia em até 11% da classificação quando comparada às CNNs tradicionais do estado da arte.

Palavras Chave: Redes Neurais Convolucionais; Detecção de Pornografia em Vídeos;

Abstract

The task of pornographic video detection has been the subject of several studies in the literature. In general, a video is divided into frames, which are then sorted separately. However, although such an approach is widely studied, it is not able to detect sensitive context-dependent content. For example, according to the context, an unclothed person may be taking a shower or participating in a sexual-related scene. In light of this, this work proposes a resource-friendly approach for the detection of sensitive content in videos. In the proposal, an image-based CNN is trained to detect pornographic images for the classification of individual video frames. Then, a pattern recognition algorithm, fed with CNN outputs, is applied, extracting context-dependent information in a time-window-based approach. The proposed method is evaluated in a new dataset, made of over 476 thousand individually classified images, built from more than 14 thousand videos. The evaluation show that traditional image-trained CNNs are unable to detect pornographic videos according to their context. In addition, the proposed approach improves accuracy by up to 11% when compared to traditional state-of-the-art CNNs.

Keywords: *Convolutional Neural Networks; Pornographic Video Detection;*

Lista de Ilustrações

Lista de Tabelas

Apêndices Lista de Tabelas

Lista de Abreviaturas e Siglas

2D	*Two Dimensions*
3D	*Three Dimensions*
3GP	*Third Generation Partnership*
AVG	*Average Value*
AVI	*Audio Video Interleave*
BoW-VD	*Bag of Words Video Descritor*
C/C++	*C and C++ Programming Language*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
CRT	*Class Related Thresholds*
DBN	*Deep Belief Networks*
DSN	*Deep Stacking Networks*
FC	*Fully Connected Layer*
FN	*False Negative*
FP	*False Positive*
FPS	*Frames Per Second*
GPU	*Graphics Processing Unit*
HSV	*Hue Saturation Value (color model)*
ILSVRC	*ImageNet Large-Scale Visual Recognition Challenge*
KNN	*K - Nearest Neighbors*
LLM	*Lexical Level Matching*
LSTM	*Long Short Term Memory*
MAX	*Maximum Value*
MD5	*Message Digest Algorithm 5*
MFCC	*Mel Frequency Cepstral Coefficients*
MIN	*Minimum Value*
MP4	*MPEG-4 Video (Moving Picture Experts Group – 4)*
MSD	*Multi-Granular Score Distribution*

RBM	*Restricted Boltzmann Machine*
RCNN	*Region Based Convolution Neural Network*
RGB	*Red Green Blue (color model)*
ROI	*Regions Of Interest*
RQ	*Research Question*
RNA	*Artificial Neural Networks*
SGD	*Stochastic Gradient Descent*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded Up Robust Features*
SVM	*Support Vector Machines*
TP	*True Positive*
TN	*True Negative*
OpenCV	*Open Source Computer Vision Library*
PCA	*Principal Component Analysis*
PROS	*Prosodic Features*
WEKA	*Waikato Environment for Knowledge Analysis*
YCbCr	*Luminance; Chroma: Blue; Chroma: Red*
YIQ	*Color Space Used by the NTSC color TV system*
YUV	*Color Space Used by the PAL color TV system*

Lista de Notações Matemáticas

$\sum$	Função somatória
N	Tamanho total da janela de *frames*
F_b	*Frame* base
x_i	*Frame* na posição corrente
$P_{internal}$	Percentual de *frames* pornográficos sobre o tamanho total da janela de *frames*
$P_{previous}$	Percentual de *frames* pornográficos localizados da janela anterior
P_{Next}	Percentual de *frames* pornográficos localizados da janela posterior
$T_{previous}$	Posições de *frames* deslocados do *frame* base retrocedendo a janela até o próximo *frame* pornográfico
T_{next}	Posições de *frames* deslocados do frame base avançando até o próximo *frame* pornográfico
Q_{frame}	Posições de deslocamento
$f(x)$	Função de x
F_β	Valor de *F-Measure* em função de β
β^2	Valor de β elevado a potência de dois
β	Peso de ponderação do cálculo de *F-Measure* adotado por Perez et al.
$<$	É menor que
$\geq$	É maior ou igual a
σ	É o desvio padrão
μ	É a média da população
X	É o valor do elemento
z	É o valor de z-score

Sumário

Capítulo 1

Introdução

Recentemente, com a crescente adoção das novas tecnologias digitais de informação e comunicação, o acesso de crianças e adolescentes a dispositivos móveis (*smartphones* e *tablets)* tem se tornado cada dia mais precoce (Collera, 2019). Esse fator gera preocupação dos pais em relação à utilização dos dispositivos móveis de maneira "inteligente". Existe uma grande variedade de conteúdos e nenhuma classificação dos mesmos para ajudar o controle paterno (Giuseppe, et al., 2009), a fim de auxiliar os pais/responsáveis a monitorar o que os jovens podem acessar e postar nas redes sociais/youtube.

Dado esse cenário, diversos especialistas têm estudado e alertado os pais em relação aos perigos associados à utilização de aparelhos móveis (*smartphones* e *tablets*). Dias et al. afirmam que o uso frequente desses dispositivos móveis limita o desenvolvimento físico, mental e psicológico das crianças (Dias, et al., 2015). Além dos problemas constatados por Dias, existe uma ameaça que pode ser ainda maior: a exposição dos jovens aos riscos e perigos da Internet. Portanto, torna-se necessário ter cautela em relação tanto ao conteúdo que pode ser visualizado, quanto ao conteúdo transmitido pelos aparelhos. Boa parte dos conteúdos disponibilizados na Internet não são adequados para crianças e adolescentes, que acabam sendo expostos a conteúdos de caráter ofensivo, violento, pornográfico ou ainda, uma mistura desses. Conforme os dados estatísticos, é estimado que cerca de 30% da proporção do conteúdo da Internet corresponde a material de teor pornográfico (Yagielowicz, 2012) (Enough, 2019).

Esse tipo de conteúdo sensível pode ser exposto em diferentes mídias, sendo o formato de vídeo atualmente o mais comum (Cybersecurity, 2019). Diversas abordagens para detecção de conteúdos sensíveis em vídeos foram propostas na literatura (Jansohn, et al., 2009) (Perez, et al., 2017), entretanto inibir a transmissão do conteúdo ofensivo nos dispositivos móveis ainda é uma tarefa extremamente complexa, principalmente quando presente em formato de vídeo. Na classificação de uma imagem, devido sua propriedade

estática, um algoritmo analisa cada um de seus *pixels* e lhe atribui um rótulo. Contudo, a tarefa de classificação de um vídeo torna-se mais complexa em virtude da transição de *frames* (cenas dos vídeos), pois um mesmo vídeo pode conter trechos que contenham pornografia e outros não. Assim, para classificar um vídeo é necessário avaliar cada um dos seus *frames*.

Visando a proteção de menores de idade, ferramentas foram desenvolvidas para efetuar um controle de acesso a conteúdo impróprio. Entretanto, a maior parte das abordagens utilizadas, limitam-se apenas na filtragem do conteúdo impróprio que é visualizado no *browser* dos aparelhos. Tal estratégia é válida, entretanto, é importante destacar que existem diversos aplicativos que utilizam a Internet e assim são introduzidas brechas no que se refere a disseminação de conteúdo ofensivo, principalmente quando relacionado a conteúdo pornográfico. Existe uma grande dificuldade em relação a filtragem do conteúdo que é disponibilizado pelos aplicativos proprietários, tais como Whatsapp[1], Facebook[2], Instagram[3], entre outros.

Apesar de existirem políticas de uso e privacidade dos aplicativos, pessoas que estejam mal-intencionadas farão uso de dispositivos de maneira ilícita, independentemente de suas políticas. As práticas ilegais na Internet são crescentes, onde criminosos tomados por uma sensação de impunidade e uma certa ilusão do anonimato, realizam crimes cibernéticos (Morais, 2018). Entre essas práticas se enquadra o crime de pedofilia, caracterizado pela troca de informações ou imagens pornográficas que envolvam crianças/adolescentes que sejam menores de idade (BRASIL, 2008).

Atualmente, com adoção prematura da utilização dos dispositivos móveis pelas crianças, a ingenuidade delas tem sido explorada e acabam se tornando alvos fáceis. Os *smartphones* são utilizados para práticas criminosas, quando por meio das câmeras digitais dos aparelhos, são realizados a captura e o compartilhamento de imagens e vídeos pornográficos envolvendo crianças ou adolescentes. Compete aos pais estarem atentos, para proteger as crianças e adolescentes para que esses façam um bom uso da Internet, e por isso ferramentas para auxiliar o controle paterno são imprescindíveis.

[1] https://www.whatsapp.com/
[2] https://www.facebook.com/
[3] https://www.instagram.com

1.1. Motivação

Os vídeos são amplamente utilizados para difundir pornografia, e essa mídia, devido seu caráter dinâmico, é um dos formatos mais complexos de serem classificados automaticamente. Disposto na literatura, existem inúmeros trabalhos para identificação de conteúdo pornográfico em imagens. Porém, em relação à detecção de conteúdos pornográficos em vídeos ainda existem poucos trabalhos, principalmente que abordem o problema de contexto nos vídeos.

Uma das formas de utilizar o contexto do vídeo na classificação é fornecida pelas convoluções 3D e CNN LSTM. Essas abordagens foram projetadas para capturar informações de movimento que são codificadas em múltiplos quadros adjacentes. Tais abordagens tratam o problema de contexto do vídeo utilizando *Convolutional Neural Network* (CNN) (Ji, et al., 2013). Entretanto, surge um outro problema: essas abordagens são extremamentes complexas, exigem muito recurso computacional, o que acaba inviabilizando a adoção desse tipo de abordagem em dispositivos com recursos limitados (dispositivos móveis).

A motivação desse trabalho surge a partir da limitação mencionada anteriormente, ou seja, devido à dificuldade na classificação de vídeos, principalmente quando o foco de implementação são os dispositivos móveis. Nesse contexto, para a classificação de vídeos pornográficos, torna-se necessário verificar uma sequência de *frames* e é importante salientar que nem todos os *frames* são de caráter pornográfico. Por exemplo, uma cena com nudez explícita pode ou não ser de caráter pornográfico, e.g. uma praia de nudismo. Por outro lado, um filme pornográfico não possui todos os seus *frames* com atitudes de cunho pornográfico. Outro aspecto a ser considerado, é a classificação do vídeo em tempo real em uma transmissão de vídeo ao vivo. Nesse caso, é impraticável enviar todos os *frames* para depois analisar se cada *frame* é pornográfico ou não.

Desse modo, quando os *frames* de um vídeo são analisados de maneira individual, definir corretamente se ele é ou não pornográfico se torna um desafio. Portanto, em uma avaliação que é realizada em um único frame talvez não exista informações suficientes para determinar corretamente o contexto da cena de vídeo. Baseado nessa premissa foi estabelecida a hipótese deste trabalho, ou seja, “a classificação do contexto do vídeo por meio da avaliação de uma janela de *frames* é mais eficaz que a análise de apenas um único frame”.

1.2. Objetivos

O objetivo geral do trabalho é desenvolver uma nova abordagem para detecção de pornografia (conteúdo sensível) em vídeos, em tempo real, para dispositivos com recursos limitados. Para tanto, o trabalho consiste em desenvolver uma abordagem baseada em CNN associada a uma janela de *frames* deslizante. Além disso, o trabalho objetiva modificar a topologia de uma arquitetura CNN tradicional, obtendo um modelo destinado a dispositivos com recursos limitados. Propondo uma alternativa de baixo custo computacional na tarefa de detecção de pornografia em vídeos observando o seu contexto, visando assim a redução das taxas de FP (Falso Positivo) e FN (Falso Negativo).

Para alcançar o objetivo geral do trabalho, os seguintes objetivos específicos devem ser alcançados:

1. Criar uma base de dados rotulada com vídeos de conteúdo pornográfico para treinamento e avaliação dos métodos desenvolvidos ao longo do trabalho.
2. Gerar os modelos das diferentes arquiteturas CNN utilizadas na proposta, resultantes das etapas de treinamento, validação e teste.
3. Elaborar um método para detecção de conteúdo pornográfico observando seu contexto, estruturado a partir da predição de uma sequência de *frames* advindos da predição da arquitetura CNN.
4. Definir o conjunto de atributos que sejam representativos para avaliar o contexto do vídeo por meio de uma sequência de *frames*.
5. Selecionar empiricamente os melhores classificadores a serem utilizados na abordagem proposta.
6. Definir estratégias para modificar a topologia de uma arquitetura CNN do estado da arte, focando a redução da complexidade da arquitetura, minimizando assim o custo computacional necessário para aplicação em dispositivos com recurso limitados.
7. Definir estratégias para otimização da abordagem proposta aplicada na detecção de conteúdo de vídeo em tempo real, focando o aumento do *throughput*, a redução na demanda computacional e a preservação da acurácia.
8. Aplicar a abordagem proposta sobre um dispositivo com recurso limitado.

9. Aprimorar a precisão da abordagem proposta utilizando um método verificador, observando a confiança da saída da CNN.
10. Comparar a abordagem proposta sobre um *dataset* do estado da arte.

1.3. Contribuições

A principal contribuição deste trabalho é a elaboração de um método para auxiliar a detecção de conteúdo pornográfico de vídeos em tempo real considerando seu contexto, abordagem computacionalmente leve, destinada a dispositivos com recursos limitados. Outras contribuições secundárias são elencadas abaixo:

(i) um novo *dataset* para detecção de conteúdos pornográficos, gerado a partir da extração dos *frames* dos vídeos, rotulado *frame* a *frame*, a fim de superar o desafio na classificação dos *datasets* já existentes – observando a granularidade do *dataset* rotulado por vídeo, existem vídeos pornográfico que nem todos os trechos dos vídeos contêm teor pornográfico.

(ii) uma estratégia para representar o contexto do vídeo, baseada na premissa de que uma sequência de *frames* pode transmitir informações adjacentes para interpretação e análise do contexto da cena;

(iii) uma nova abordagem de classificação baseada em duas etapas, inicialmente utilizando CNN e posteriormente uma etapa adicional baseada em janelas de *frames* - como uma alternativa de baixo custo (*light*) para detecção de pornografia considerando o contexto do vídeo;

(iv) uma nova arquitetura CNN destinada a dispositivos com recursos limitados - alterando a topologia da arquitetura CNN é possível reduzir o custo computacional necessário para rodar o modelo preservando sua precisão;

(v) um método para otimização da abordagem de janela proposta - estabelecendo estratégias adequadas para reduzir as demandas de memória, aumentar o *throughput* na detecção e melhorar a acurácia da detecção; e

(vi) avaliação do impacto da utilização do modelo proposto em um dispositivo com recursos limitados.

1.4. Organização do Documento

O documento encontra-se organizado da seguinte forma: No Capítulo 1 é realizado a introdução da proposta. O Capítulo 2 aborda a fundamentação. No Capítulo 3 é apresentado os trabalhos relacionados com a proposta. O Capítulo 4 detalha a metodologia utilizada para a realização do projeto. No Capítulo 5 é apresentado a análise dos resultados. E por fim no Capítulo 6 é apresentado a conclusão.

Capítulo 2

Fundamentação Teórica

Esse capítulo apresenta conceitos relacionados as tarefas de processamento de vídeo, extração de características e de aprendizagem de máquina, contemplando a utilização de CNN e indutores menos complexos.

2.1. Processamento de Imagens e Vídeos

De maneira geral, para as tarefas de processamento de vídeo, referente a captura e digitalização dos vídeos alguns aspectos são fundamentais, tais como resolução do vídeo, espaço de cor e taxa de *frames* (quadros). A resolução é uma característica que define as dimensões do vídeo, partindo de um plano cartesiano na posição horizontal os *pixels* são estruturados em colunas, bem como na posição vertical são organizados em linhas. Em relação ao espaço de cor, sua propriedade refere-se à quantidade de *bits* necessários para representar cada uma das cores básicas no espaço de cor, neste trabalho em específico o espaço de cor RGB (*Red*, *Green* e *Blue*). Taxa de *frames* corresponde ao número de *frames* que são exibidos por segundo, em geral, uma taxa de 23,97 FPS (*Frames Per Second*) é utilizada para se adequar a percepção humana (Nasiri, et al., 2015).

Tais características são abstraídas pelas CNN, que recebem como entrada os *frames* dos vídeos como se fossem imagens. Portanto, para permitir o processamento computacional dos vídeos utilizando a CNN, torna-se necessário efetuar a extração dos *frames*.

2.1.1. Detecção de Conteúdo em Vídeos

A detecção de conteúdo de vídeo é normalmente obtida por meio de dois módulos sequenciais, extração de *frames* e classificação de *frames*, como demostrado na

Figura 1 (Detecção de conteúdo de vídeo). Iniciamente na extração de *frames*, o vídeo é dividido em um conjunto de *frames*. O número de *frames* é definido de acordo com o processo de codificação do vídeo. Por exemplo, em geral, um vídeo é feito de 23,97 *frames*/seg. (Kuroki, 2007). Então, no processo de classificação do frame, cada frame extraído é classificado individualmente. Durante esta fase, é atribuído um rótulo para cada *frame*, realizada por meio de uma técnica de classificação, como a adoção do uso de uma CNN. Finalmente, de acordo com o rótulo atribuído é possível executar algum tipo de ação, por exemplo alertar o usuário.

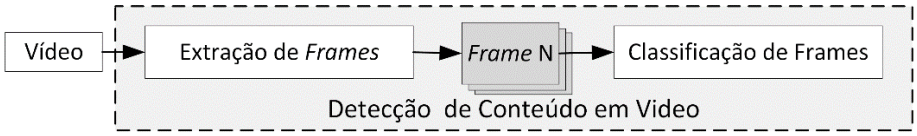

Figura 1 - Típico processo de detecção e conteúdo em vídeo

Em um cenário direcionado a classificação do conteúdo de vídeo em tempo real, os *frames* devem ser classificados na velocidade em que o conteúdo do vídeo é transmitido. Assim, o processo de detecção deve ter uma taxa de transferência de detecção maior que a taxa de *frames* de vídeo (por exemplo, maior que 23,97 *frames/seg*.). Visando executar a tarefa de detecção de conteúdo de vídeo em dispositivos com recursos limitados de hardware, tal como um *smartphone*, será necessário classificar cada *frame* individualmente, além de executar outras tarefas em paralelo, como reproduzir o vídeo ou executar outros aplicativos em segundo plano. Consequentemente, a tarefa de detecção de conteúdo de vídeo deve utilizar o mínimo de recursos possível. Observando que a maior demanda de execução pela tarefa de detecção do conteúdo do vídeo é principalmente ocasionada pelo módulo de classificação de *frames*. Isso porque esse módulo deve avaliar cada um dos *frames* extraídos dos vídeos e rotulá-los adequadamente.

2.2. Aprendizagem de Máquina

Aprendizagem de máquina situado na área de Inteligência Artificial, é o campo do estudo que explora os métodos de análise de dados que permitem automatizar o desenvolvimento de modelos analíticos, por meio de algoritmos que apreendam interativamente um modelo a partir de seus dados (Wedemann, 2019). Assim

possibilitando a compreensão de maneira autônoma, permitindo que os algoritmos efetuem predições ou ainda tomem decisões.

Esse aprendizado é baseado na observação dos dados, reconhecendo e extraindo padrões dos dados, geralmente quanto mais exemplos forem avaliados, melhor será a capacidade de generalização do modelo. Uma vez que o modelo tenha aprendido, esse será capaz de executar tarefas complexas e dinâmicas. Em comparação com os algoritmos tradicionais, a implementação de algoritmos que aprendam a partir de exemplos é uma tarefa mais complexa, e exige um custo adicional de processamento em relação a implementação tradicional (Domingos, 2012). Como exemplo, podemos citar as tarefas complexas de cognição, tais como reconhecimento de objetos e processamento de linguagem natural (Lecun, et al., 2015) (Deng, et al., 2013). Pesquisas recentes têm demonstrado que as redes de aprendizagem profunda têm superado os algoritmos tradicionais, especialmente nas tarefas de visão computacional e reconhecimento de padrões, tal como classificação de objetos em imagens (He, et al., 2016).

2.2.1. Classificadores

Aprendizagem de máquina supervisionada pode ser definida como sendo um processo a fim de apreender um conjunto de regras, a partir de certas instâncias ou exemplos rotulados de um conjunto de treinamento (classificação e regressão). Ainda pode ser caracterizado, como sendo um modelo estabelecido por um classificador, capaz de generalizar novas instâncias que não estão contidas no conjunto de treinamento. Conforme a definição de Kotsiantis et al., classificador é um modelo que após treinado pode ser utilizado para definir classes a instâncias de testes, cujas classes são desconhecidas, utilizando a informação dos seus atributos (Kotsiantis, et al., 2007). Na classificação, o modelo aprendido tem saída discreta (categórica), enquanto na regressão a saída é um valor contínuo. Existem diferentes tipos de indutores, tais como redes neurais artificiais, redes bayesianas, árvores de decisão, máquina de vetores de suporte, entre outros. Nas próximas subseções serão descritos os principais classificadores que foram utilizados neste trabalho.

2.2.1.1. Algoritmo Naive Bayes

O classificador Naive Bayes é um algoritmo baseado no Teorema de Bayes proposto por Thomas Bayes (1701-1761). Esse é um classificador probabilístico que assume que as características são condicionalmente independentes, ou seja, as ocorrências de uma determinada característica não afetam a probabilidade da ocorrência de outra característica, desconsiderando completamente a correlação entre variáveis (Zhang, 2004). Assim, o modelo consiste em uma tabela de probabilidades para cada característica, obtidas a partir da base de treinamento. Na classificação, a probabilidade *a priori* de cada classe não é dependente das outras características, sendo assim, essa é multiplicada pela probabilidade individual de cada característica. Onde a classe com maior probabilidade é atribuída ao evento a ser classificado.

2.2.1.2. Algoritmo C4.5

O algoritmo C4.5 foi introduzido por Quinlan para indução de modelos de classificação (Quilan, 1993), frequentemente conhecido por ser uma árvore de decisão. As árvores de decisão geradas pelo C4.5 são também utilizadas para classificação e por isso esse algoritmo muitas vezes acaba sendo referenciado como classificador estatístico. A árvore de decisão é um algoritmo de classificação que consiste em percorrer uma árvore da raiz até chegar a um nó folha, associando-se a uma determinada classe. Cada nó interno dessa árvore estabelece um teste de uma característica (regras condicionais), e o ramo correspondente refere-se a um valor possível da característica. Uma implementação Java do algoritmo C4.5 é o J48, disponibilizado para ferramenta de mineração de dados WEKA[4] (Weka, 2019).

2.2.1.3. Algoritmo KNN

O classificador KNN não cria exatamente um modelo explícito, esse efetua a classificação a partir da base de treinamento. Sendo os eventos classificados baseados na similaridade entre as instâncias vizinhas. Durante o processo de classificação, é definido

[4] https://www.cs.waikato.ac.nz/ml/weka/

a variável K, correspondente a quantidade de vizinhos que são determinados por uma métrica de distância, assim como a distância euclidiana. Observando o espaço de características das instâncias entre as classes, é determinada a classe que possui maior similaridade. A classe com maior número de ocorrência entre os K vizinhos é selecionada como classe da instância sendo classificada. Dado que o KNN deve calcular a distância entre o evento com todas as instâncias dispostas na base de treinamento, quanto maior for o número de instâncias na base de treinamento, maior será o tempo para o processamento. Outro aspecto que deve ser levado em consideração, as características com um valor absoluto tendem a sofrer influência no cálculo da distância, assim essas características devem ser normalizadas.

2.2.1.4. Algoritmo Random Forest

O algoritmo *Random Forest*, publicado por Breinen em 2001, refere-se a uma abordagem conjunta para a construção de modelos preditivos. A "floresta", nessa abordagem, é uma série de árvores de decisão que atuam como classificadores "fracos" quando utilizados individualmente, porém, em conjunto, formam uma previsão robusta (Breiman, 2001). São utilizados em diferentes domínios devido a sua natureza simples e seu alto desempenho. O *Random Forest* não realiza inferências fortes sobre a escala e normalidade do conjunto de dados. Trabalham bem com dados numéricos e categóricos mistos, e não exigem muito ajustes para obter uma versão preliminar razoável de um modelo preditivo. Além dos algoritmos serem rápidos de treinar e intuitivos, fornecem importância de recursos como uma característica do modelo, são inerentemente capazes de lidar com dados ausentes (Storey, 2018). Tais características tornam o *Random Forest* um interessante ponto de partida para explorar novos domínios ou criar um modelo preditivo.

2.2.1.5. Algoritmo Adaboost

O algoritmo AdaBoost foi proposto por Freund e Schapire em 1995. O AdaBoost é um algoritmo meta-heurístico que pode ser utilizado também para aperfeiçoar o desempenho de outros algoritmos de aprendizagem. O AdaBoost é um algoritmo adaptável, onde as classificações subsequentes são ajustadas conforme as instâncias classificadas anteriormente. Esse algoritmo é sensível ao ruído nos dados e em casos

isolados. Porém para alguns problemas é menos suscetível à perda da capacidade de generalização após o aprendizado de muitos padrões de treino (*overfitting*) do que a maioria dos outros algoritmos (Freund & Schapire, 1997). O AdaBoost chama um classificador de menor potencial repetidamente em iterações $t = 1, \dots, T$. Onde para cada chamada à distribuição dos pesos D_t é atualizada para indicar a importância do exemplo no conjunto de dados. Em cada iteração os pesos de cada exemplo classificado erroneamente são ampliados, forçando que um novo classificador foque um pouco mais nesses exemplos.

2.2.2. Redes Neurais Artificiais

Uma rede neural artificial tradicional é composta pela ligação de diversos neurônios, estruturada a partir de uma sequência de ativações. A aprendizagem na rede é obtida ao encontrar a correlação entre os pesos, permitindo que sejam realizadas ativações apropriadas. Em algumas circunstâncias uma única camada talvez não seja autossuficiente para representar os dados de entrada.

Visando atender a tal demanda foram propostas as redes de aprendizagem profunda, em inglês denominadas como *Deep Learning* (DL). Conforme os autores (Lecun, et al., 2015), *Deep Learning* é definido como sendo métodos que permitem que modelo composto de várias camadas aprendam representações dos dados em diversos níveis de abstrações. Esses algoritmos mostraram serem capazes de superar os algoritmos tradicionais, chegando muito próximos do desempenho do nível humano e, em alguns casos, até mesmo sendo superiores. O *Deep Learning* refere-se a um subgrupo das RNAs (*Artificial Neural Networks*), que são redes compostas por múltiplas camadas (Schmidhuber, 2015). As RNAs já existiam por décadas, porém não havia recurso computacional suficiente para treinar tais arquiteturas, permanecendo assim até meados dos anos 2000. Alguns fatores contribuíram no recente aumento do uso de aprendizagem profunda: inovações dos algoritmos - inicialmente proposta por George Hinton; o aumento da capacidade computacional com adoção das GPUs (*Graphics Processing Unit*) e a utilização de conjuntos maiores de dados.

As Arquiteturas de *Deep Learning* geralmente são muito complexas, sendo um tipo de aprendizagem de máquina que treina computadores para realizar tarefas semelhantes aos seres humanos. Tais arquiteturas atuam em áreas distintas, tais como percepção, raciocínio e aprendizagem. Assim, possibilitando realizar tarefas como

reconhecimento (objetos, texto, fala, cenas, faces, entre outras) (He, et al., 2016) (Irsoy & Cardie, 2014) (Deng, et al., 2013), classificação de imagem e previsões (como cadeias de DNA) (Alipanahi, et al., 2015).

Entre os métodos *Deep Learning* existentes, dispostos na literatura temos CNN, RCNN, LSTM, DBN e o DSN. Segue abaixo uma descrição sucinta dos métodos:

- CNN (*Convolutional Neural Network*): é o método *Deep Learning* mais comumente utilizado, esse método tenta dividir o problema original de maneira similar ao cérebro humano (Fukushima, 1980).
- RCNN (*Region Based Convolutional Neural Network*): algoritmo que possibilita uma ligação entre os dados de entrada, realizando a classificação a partir de diversos segmentos distintos da entrada no mesmo instante (Irsoy & Cardie, 2014).
- LSTM (*Long Short Term Memory*): é uma arquitetura de aprendizagem profunda capaz de utilizar as informações armazenadas a longo prazo para o treinamento, adotando uma célula de memória (Hochreiter & Schmidhuber, 1997).
- DBN (*Deep Belief Networks*): é uma rede multicamadas onde cada par de camadas conectadas é uma máquina Boltzmann (RBM) restrita, sendo representada como uma pilha de RBMs (Jones, 2017).
- DSN (*Deep Stacking Networks*): se difere um pouco das estruturas de aprendizado profunda convencionais, sendo esta constituída por um conjunto profundo de redes individuais, cada uma com suas próprias camadas ocultas (Jones, 2017).

Atualmente já existem inúmeras aplicações que utilizam *Deep Learning* e são inúmeras as possibilidades a serem exploradas. Os pesquisadores constantemente vêm apresentando novos algoritmos e arquiteturas, ampliando consideravelmente o potencial do *Deep Learning*. Nas seções subsequentes é transcrito um pouco sobre as redes CNN, principais camadas, arquiteturas, treinamento e classificação das CNNs, por fim como é possível otimizar o modelo por meio do ajuste de parâmetros e adoção do *Transfer Learning*.

2.2.3. CNN - Redes Neurais Convolucionais

As redes CNN são redes neurais voltadas geralmente para reconhecimento e classificação de imagens. As redes neurais convolucionais (CNNs) têm alcançado resultados impressionantes em tarefas desafiadores tais como reconhecimento de imagem e detecção de objetos (Krizhevsky & Hinton, 2009) (Krizhevsky, et al., 2012) (Ciregan, et al., 2012), o que tem ampliado significativamente o interesse dos pesquisadores em tais métodos.

Semelhantes as redes neurais tradicionais, as CNNs também são constituídas por neurônios, nos quais são atribuídos pesos e viés. Cada neurônio recebe valores de entrada, e internamente são aplicados o produto escalar e uma função não linear. À medida que a rede é treinada, os pesos podem ser recalculados, geralmente utilizando a técnica de *backpropagation*, e assim a rede consegue aprender.

Umas das principais características da CNN é a sua capacidade de ser escalável (Strigl, et al., 2010), sendo ideal para tarefas de visão computacional, pois isso permite processar uma grande quantidade de informação de entrada advindas dos *pixels* das imagens ou dos *frames* dos vídeos. Os neurônios são agrupados em camadas, e por sua vez a arquitetura CNN é constituída por essas camadas. Entre as principais camadas, temos a camada de entrada sendo responsável em absorver a informação dos *pixels* convolucional onde são definidos os filtros; *pooling* que permite reduzir o número de características e totalmente conectada que determina a probabilidade e saída da classe. O conjunto das diferentes camadas permitem extrair e codificar certas propriedades da imagem, a estruturação desse conjunto de camadas é o que chamamos de arquitetura CNN. Na Figura 2 extraída do artigo (LeCun, et al., 1998) é apresentada a arquitetura LeNet-5, bastante ressaltada na literatura, utilizada em problema típico de reconhecimento de dígitos.

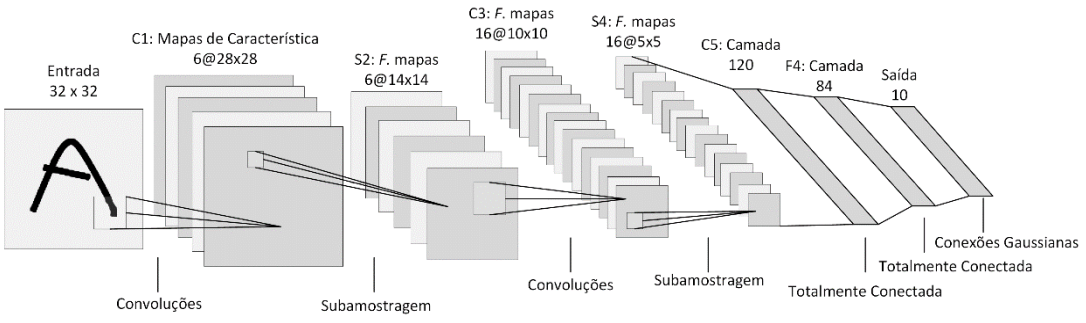

Figura 2 - Arquitetura CNN LeNet-5 adaptada de Yann LeCun (LeCun, et al., 1998)

O treinamento de uma rede exige tempo e apresenta um alto custo computacional, visando otimizar o treinamento, tem se utilizado o *Transfer Learning*. *Transfer Learning* é um método que permite reutilizar um modelo que foi gerado anteriormente para determinado fim, e reaproveitar os pesos como ponto de partida para gerar um novo modelo.

Abordagens com adoção das CNNs são crescentes e vem sendo exploradas nos mais diferentes âmbitos. Principalmente devido ao fato de as CNNs terem comprovado sua eficiência em termos de acurácia aplicadas em tarefas de reconhecimento de padrões e visão computacional. Tradicionalmente as CNNs recebem como entrada um conjunto de imagens, recentes propostas introduzem as CNNs em tarefas de classificação de vídeo.

Na classificação de vídeos o enfoque é capturar e codificar as informações de cunho dinâmico que são obtidas dos *frames* dos vídeos. Assim, a CNN recebe como entrada uma sequência *frames*, extrai as características de cada um dos *frames*, de maneira similar ao processo realizado com imagens. No treinamento, a CNN absorve informações advindas de cada um dos *frames*, desse modo a rede começa a "aprender". Ou seja, a CNN abstrai padrões encontrados nos *frames*, posteriormente isso permite determinar sua classe. Dessa maneira, torna-se possível capturar informações de movimento que são codificadas em *frames* adjacentes.

Visando compreender melhor o funcionamento da CNN, as subsecções seguintes expõem o funcionamento das principais estruturas e são apresentados com maiores detalhes os diferentes tipos de camadas, filtros e funções de ativação.

2.2.3.1. Camada Conv (Convolucional)

Esta camada consiste em um conjunto de filtros aprendíveis que são deslizados sobre a imagem de forma espacial. Assim a camada convolucional calcula a saída dos neurônios conectados a regiões locais na entrada, cada um computando um produto de ponto entre seus pesos e uma pequena região a qual estão conectados (Karpathy & Johnson, 2018). Os filtros devem percorrer a profundidade total da imagem de entrada. Para exemplificar, caso seja desejável aplicar um filtro de tamanho 5x5 sobre uma imagem colorida de tamanho 32x32, o filtro deverá ter profundidade 3 (5x5x3) para ser possível cobrir todos os 3 canais do espaço de cor RGB (*Red Green Blue*) da imagem (Moujahid, 2016).

Na Figura 3 exposta por Karpathy, é apresentado no bloco à esquerda um exemplo de volume de entrada (disposto nas dimensões 32x32x3), no bloco à direita um exemplo de volume de neurônios na primeira camada Convolucional. Aqui cada neurônio na camada convolucional está conectado apenas a uma região local no volume de entrada espacial, na mesma profundidade, ou seja, conectados em todos os canais de cores. Observe a existência de múltiplos neurônios, sendo representado 5 neste exemplo, ao longo da profundidade, todos focados na mesma região na entrada (Karpathy & Johnson, 2018).

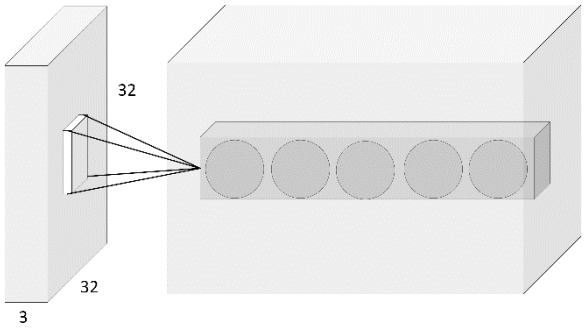

Figura 3 - Representação da camada convolucional por Karpathy adaptada (Karpathy & Johnson, 2018)

2.2.3.2. Camada de Pooling

A camada de *Pooling* foi projetada para reduzir o tamanho espacial da representação da imagem, diminuindo assim suas características. Efetuando a redução da quantidade de parâmetros e computação na rede é possível controlar o *overfitting*. Essa camada trabalha de forma independente em cada pedaço da profundidade da entrada, no redimensionando da imagem utiliza uma função de agrupamento não linear.

Existem diversas funções para implementar o agrupamento, sendo o conjunto máximo a mais utilizada. A forma mais comum é uma camada de agrupamento com filtros de tamanho 2x2 aplicados com um passo de 2 amostras em cada fatia de profundidade (na entrada de 2 ao longo da largura e altura), que encolhe a imagem de entrada para 1/4 do tamanho original, descartando assim 75% das ativações (Karpathy & Johnson, 2018). Nesse caso, cada operação *MAX pooling* levaria no máximo 4 números, considerando a região 2x2 em alguma fatia de profundidade, permanecendo sua dimensão inalterada. Na Figura 4 é apresentada a camada de *pooling* esquematizada por Karpaty:

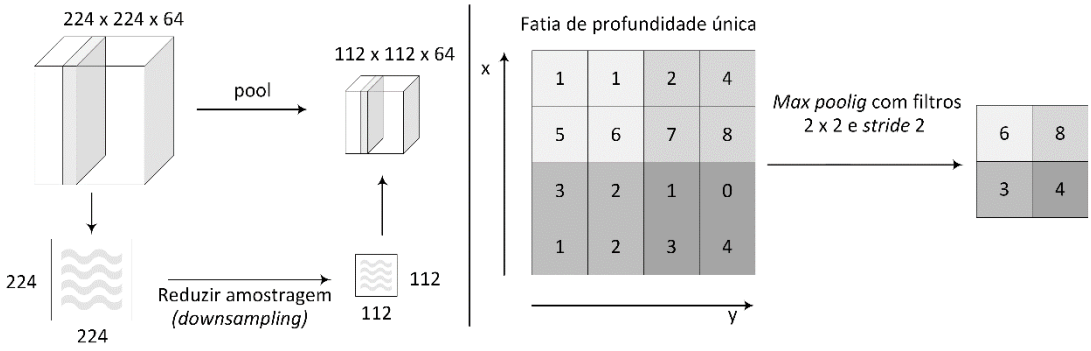

Figura 4 - Representação adaptada da camada de pooling por Karpathy (Karpathy & Johnson, 2018)

Podemos observar a partir da imagem que a camada de agrupamento reduz a amostragem do volume espacialmente, independentemente e em cada fatia de profundidade do volume de entrada. A lado esquerdo, os autores expõem um volume de entrada de tamanho [224x224x64] sendo agrupado com o tamanho 2 de filtro, passo 2 no volume de saída de tamanho [112x112x64], em que a profundidade do volume é preservada. A lado direito Karpaty é apresentada a operação de MAX *pooling*, dando origem a pool máximo, aqui mostrado com um passo de 2, sendo que cada MAX é preenchido com 4 números (quadrado de 2x2).

Em outros termos, o MAX *pooling* é um filtro que é aplicado sobre toda extensão da imagem que verifica qual o maior valor encontrado em cada um dos segmentos, reservando o valor máximo desse segmento. Após o filtro ser aplicado em toda imagem de entrada, temos uma nova imagem com sua dimensão e características reduzidas. Existem alternativas ao MAX *pooling*, tais como o MIN *pooling* e o AVG *pooling*. O MIN *pooling* semelhante ao MAX *pooling* é aplicado sobre toda a extensão da imagem, porém essa busca pelo menor valor em cada um dos segmentos resulta em uma nova imagem com menor brilho. Por sua vez, AVG *pooling* calcula o valor médio para cada um dos segmentos, e a aplicação desse filtro tende a suavizar a nova imagem, o que reduz a nitidez da imagem.

2.2.3.3. Função de Ativação – ReLU

A função de ativação é uma função que calcula a saída dos neurônios artificiais. A entrada que a função recebe representa a soma de todos os produtos de suas entradas e seus respectivos pesos, o que pode ser denominado como "soma ponderada" (Parfenovich, 2012). Essas funções codificam padrões complexos dos dados de forma

binária, determinando a saída da rede neural como sim ou não, sendo mapeados os valores resultantes entre 0 a 1 ou variando de -1 a 1 dependendo da função. Entre as funções de ativação mais conhecidas temos a Sigmoid, Tanh e a ReLU.

Atualmente ReLU é uma das funções de ativação mais utilizadas, sendo usadas em quase todas as redes neurais convolucionais ou de aprendizado profundo. A função ReLu calcula a função $f(x) = \max(0, x)$, conforme a Equação 1:

$$f(x) = \begin{cases} 0 \; para \; x \; < 0 \\ x \; para \; x \geq 0 \end{cases} \quad (1)$$

Em outras palavras, a ativação é simplesmente limitada a zero.

2.2.3.4. Camadas Totalmente Conectadas

A camada totalmente conectada recebe um volume de entrada das camadas posteriores (Camada de Convolucional, ReLu ou Pool) e gera um vetor de tamanho correspondente ao número de classes. Assim, o valor da saída de cada neurônio em uma camada é determinado por sua conexão ponderada com cada neurônio na camada anterior (Jacobson, et al., 2018). Ainda, cada número na dimensão do vetor refere-se à probabilidade de uma determinada classe. A representação da saída por meio da probabilidade é obtida adotando a abordagem *Softmax*, porém é necessário ressaltar a existência de outras maneiras de representar a saída da rede.

De maneira geral, a camada totalmente conectada funciona examinando a saída das camadas anteriores, assim efetua uma análise e determina quais os recursos que estão mais correlacionados com determinada classe. Geralmente a camada FC (*Fully Connected Layer*) examina quais recursos de alto nível estão mais fortemente interligadas a determinada classe. Por fim, examina seus pesos específicos para calcular os produtos entre os pesos e a camada anterior, obtendo assim as probabilidades corretas para as diferentes classes.

Na Figura 5 é apresentado um exemplo de uma camada FC adaptado do modelo proposto por Karn (Karn, 2016), correspondente a uma tarefa de classificação de imagens que possui quatro possíveis saídas. A fim de facilitar o entendimento, as conexões entre os neurônios e as possíveis saídas das classes não foram representadas na imagem, entretanto cada neurônio está diretamente interligado a todas as classes.

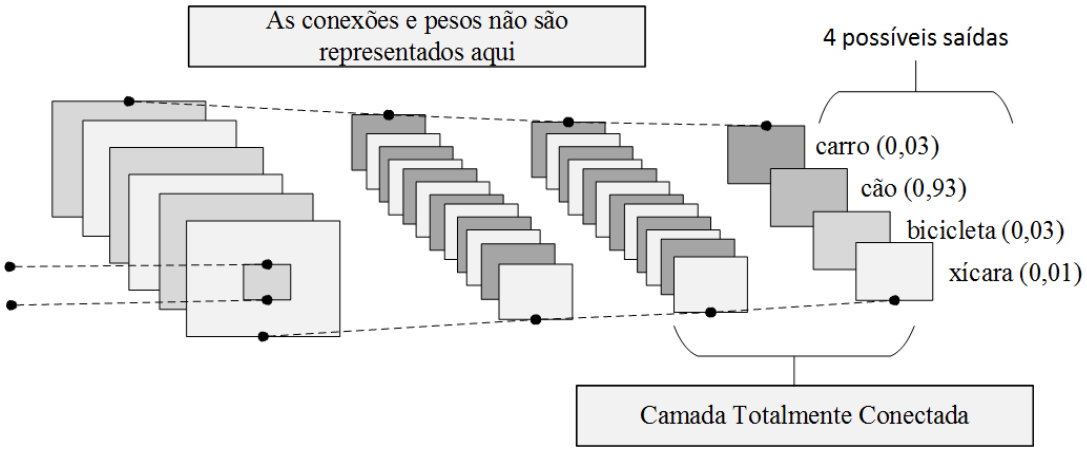

Figura 5 - *Fully connected layer* adaptado de Karn 2016 (Karn, 2016)

2.2.4. Treinamento CNN

Na fase de treinamento, a CNN tem como objetivo aprender os pesos da rede a partir dos dados de entrada, sendo que a quantidade dos dados influencia na precisão do modelo, e quanto mais dados para o treinamento, melhores serão os resultados obtidos. No treinamento de uma rede neural dois aspectos são fundamentais:

- Dados de treinamento;
- Função de perda (*loss*).

Os dados de entrada para o treinamento são compostos pelas imagens/*frames* e seus rótulos correspondentes.

2.2.4.1. Função de Perda (loss)

A Função de perda permite avaliar a imprecisão das previsões. Na fase de treinamento é necessário o ajuste de alguns parâmetros para que a rede comece efetivamente a apreender. Ressalta-se a necessidade do ajuste de um parâmetro em específico, denominado *learning rate*, esse diretamente interligado com a função de perda (*loss*). Existem diferentes funções de perda, calculadas com distintas métricas. Alguns exemplos bastante utilizados de função de perda na literatura são *MSE*, *Cross-Entropy*, *Categorical CE*, entre outros. O *MSE* calcula a diferença da média do quadrado entre os valores reais e o previsto. Por sua vez, o *Cross-Entropy* mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1, e o valor de saída é passado por uma função de ativação *Sigmoid*. Já o *Categorical CE* é uma função

de perda utilizada quando temos uma tarefa de classificação de várias classes, e a saída da camada final deve ser passada através de uma ativação *Softmax*. A função de ativação *Softmax* retorna à probabilidade de cada classe de destino, e dadas as previsões reais do modelo, valor da probabilidade que varia entre 0 e 1. Uma alternativa ao *Softmax* é a função *SoftmaxWithLoss*. Essa função calcula a perda logística multinomial para uma das muitas tarefas de classificação, passando previsões reais com uma função *Softmax* para obter a distribuição de probabilidade entre as classes.

Visando obter um valor menor para o *loss*, antes de efetivamente treinar a rede, é efetuando um pré-processamento. Para tal, é definido um valor inicial para o *learning rate*, então a rede é treinada com um número pequeno de iterações. Posteriormente, observa-se se existe redução no valor do *loss* e é efetuado o teste com diferentes valores para o *learning rate*. Esse procedimento é aplicado até alcançar um valor aceitável para o loss. Na sequência, a rede é efetivamente treinada, utilizando o valor de *learning rate* que obteve o menor valor para o *loss*.

2.2.5. Arquiteturas CNN

A utilização das CNNs provou ter bastante êxito em tarefas de visão computacional e reconhecimento de padrão, despertando ainda mais o interesse dos pesquisadores. Observando o desempenho das CNN em tais tarefas com intuito de amplificar sua eficácia, surgem novas propostas e assim, atualmente, existem gama considerável de arquiteturas CNNs. Algumas dessas arquiteturas foram utilizadas nessa proposta que seguem detalhas nas subseções posteriores.

2.2.5.1. Alexnet

A Alexnet é considerada uma das mais importantes arquiteturas de aprendizagem profunda dispostas na literatura. Foi projetada por Krizhevsky, Sutskever e Hinton visando participar da competição do ILSVRC (*ImageNet Large-Scale Visual Recognition Challenge*) de 2012. Essa competição é um dos mais importantes eventos na área de visão computacional e reconhecimento de padrão. Realizado anualmente, equipes do mundo todo competem para verificar quem possui o melhor modelo de visão computacional em tarefas tais como classificação, localização, detecção e outras. A

Alexnet no desafio da edição do ILSVRC de 2012 acabou gerando um marco nas tarefas de reconhecimento de padrão, pela primeira vez na competição uma arquitetura CNN conseguiu atingir uma taxa de erro de 5% e 15,4% sobre a base de teste (erro top-5). Destacando-se como a segunda mais bem colocada na competição, obteve o erro de 26,2%. O desempenho alcançado pela Alexnet, tornou a utilização das CNNs familiares nas versões seguintes da competição. Os autores projetaram a Alexnet com 5 camadas conv, camadas de agrupamento máximo, camadas densas e 3 camadas totalmente conectadas (Krizhevsky, et al., 2012). A Figura 6 apresenta a representação gráfica da arquitetura Alexnet projetada por Krizhevsky.

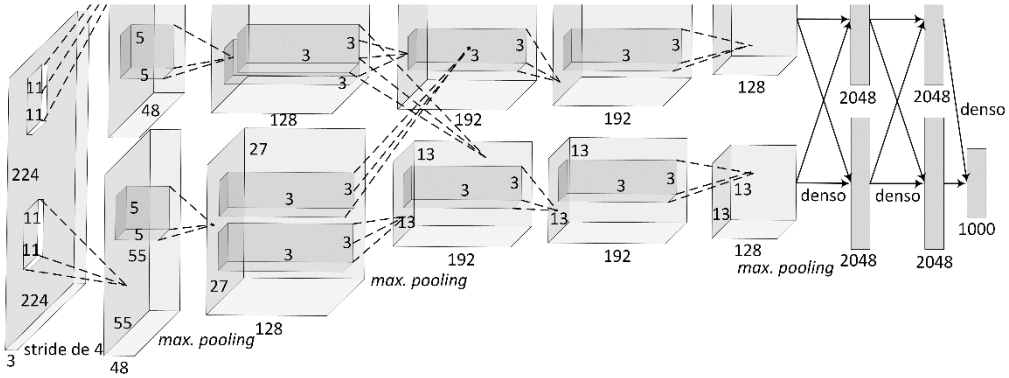

Figura 6 - Arquitetura Alexnet adaptada de Krizhevsky (Krizhevsky, et al., 2012)

2.2.5.2. Caffenet

A arquitetura do Caffenet é uma replicação da topologia do Alexnet (Krizhevsky, et al., 2012). A similaridade foi ocasionada devido a um equívoco na implementação ao tentar reproduzir a topologia do Alexnet, assim surgindo uma nova arquitetura, o Caffenet. De maneira convencional, o Caffenet também foi projetado para reconhecer as 1000 categorias de objetos dispostos no ImageNet (Russakovsky, et al., 2015).

Ambas arquiteturas diferem pelo fato de a topologia do Caffenet não utilizar reforço de dados e também devido a ordem de *pooling* e da normalização ser comutada. Sendo assim, a topologia do Caffenet também possui 8 camadas, sendo as 5 primeiras camadas convolucionais e as 3 camadas posteriores são camadas totalmente conectadas. Para entrada do Caffenet as imagens devem ser redimensionadas para 256x256 pixels. Os vetores de características apresentam 4096 dimensões, características atribuídas na sétima

camada, a camada totalmente conectada. Por fim, ressalta-se que para o cálculo da distância o Caffenet considera o espaço da Euclidiano.

2.2.5.3. Googlenet

A rede Googlenet projetada para competição do ILSVRC 2014, também conhecida como Inception, visando reduzir a complexidade computacional, apresentou uma nova proposta na estrutura das arquiteturas CNN. A estrutura da arquitetura Inception tem como ideia fundamental, considerar que a estrutura esparsa local ótima pode ser aproximada e coberta por componentes densamente prontos e disponíveis, quando se tratando de uma rede de visão computacional.

Os autores enfatizam que a principal característica dessa arquitetura é a melhor utilização dos recursos de computação dentro da rede, projetada levando em consideração o uso de memória e energia. O *design* da Inception foi cuidadosamente elaborado: essa arquitetura é composta 22 camadas, onde foram ampliadas a profundidade e a largura da rede, porém mantendo a estimativa computacional constante. Para otimizar a qualidade, as decisões arquitetônicas foram baseadas no princípio de Hebbian e na intuição do processamento em escala múltipla (Szegedy, et al., 2015). Na Figura 7 são apresentados dois módulos que compõem a Inception, estrutura proposta por Szegedy, o primeiro uma versão do Naive e a segunda representação do módulo para redução da dimensionalidade.

A Figura 7 (a) representa o módulo Inception na versão Naive, esse módulo realiza a convolução em uma entrada com três filtros de tamanhos diferentes (1x1, 3x3, 5x5). Adicionalmente, um MAX *pooling* também é realizado e as saídas são concatenadas e enviadas para o próximo módulo. A Figura 7 (b) representa o módulo Inception com redução de dimensionalidade e, visando à redução do custo computacional, o número de canais de entrada é limitado, e assim é adicionado uma convolução 1x1 extra antes das convoluções 3x3 e 5x5. Apesar da adição de uma operação extra não ser intuitiva, as convoluções 1x1 são muito mais baratas que as convoluções 5x5, e o número reduzido de canais de entrada também ajuda. Contudo, a convolução 1x1 é apenas introduzida após a camada de MAX *pooling*.

A Inception comprovou sua qualidade no contexto de detecção e classificação de objetos vencendo ao desafio da ILSVRC 2014, apresentando a melhor taxa de erro de 6,7% no top-5 na tarefa de classificação.

Com a criação da Inception, os autores demostraram que uma estruturação criativa de camadas pode levar a resultado melhores em relação ao desempenho e à eficiência computacional.

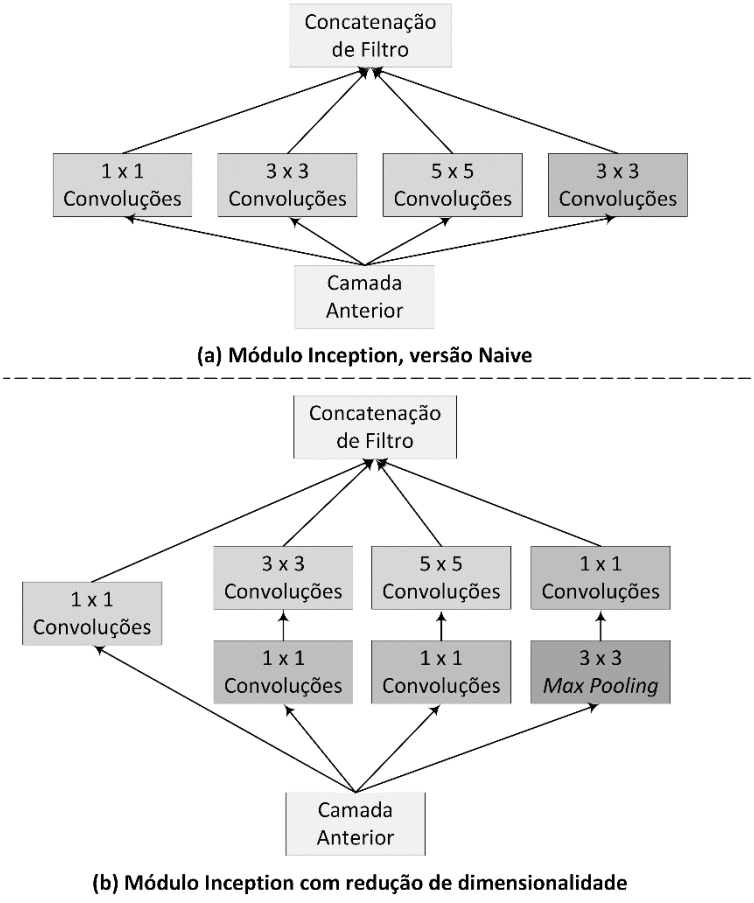

Figura 7 - Módulos Inception adaptado de Szegedy (Szegedy, et al., 2015)

2.2.6. Transfer Learning

Transfer Learning corresponde à técnica de transferência de conhecimento, onde os pesos de um modelo previamente treinado são utilizados como entrada para treinar um novo modelo focado em um contexto diferente. Ou seja, em vez de treinar a rede a partir do zero, o *Transfer learning* utiliza um modelo treinado em um conjunto de dados diferente e o adapta ao problema que estamos tentando resolver (Moujahid, 2016).

Existem duas abordagens utilizadas de maneira complementar ao *Transfer Learning*:

- **Utilizar um modelo treinado como extrator de características**: nessa abordagem é efetuada a remoção da última camada FC do modelo treinado. Os pesos das camadas

restantes são congelados, efetuando o treinamento de um classificador de aprendizado de máquina na saída dessas camadas.

- **Ajustar o modelo treinado**: nessa abordagem é efetuado o ajuste do modelo treinado no novo conjunto de dados, continuando a etapa de retropropagação. Assim é possível afinar toda a rede ou congelar algumas de suas camadas.

2.2.7. Classificação CNN

Dado um modelo CNN previamente treinado, a utilização desse novo modelo a fim de prever novas instâncias denomina-se classificação. Ou seja, a classificação refere-se à tarefa de atribuir um rótulo a uma imagem de entrada. Tal rótulo é obtido de um conjunto fixo de categorias a qual arquitetura CNN foi treinada, determinando enfim a que classe a imagem pertence.

De forma similar é possível utilizar o modelo CNN treinado para classificar uma sequência de *frames* de vídeos. Entretanto, é necessário ressaltar que o processo de classificação de imagem não é autossuficiente para extrair todas as características de um vídeo, lembrando que o vídeo possui características dinâmicas. Para o problema de detecção de pornografia em vídeos, a análise de um vídeo não depende apenas da verificação de único *frame*, o vídeo possui características próprias, diferentes da análise de uma imagem, existe o contexto no qual o vídeo está inserido.

De maneira geral, para classificação de um vídeo é intrínseco a utilização de abordagens suplementares para a rotulação dos *frames*, as abordagens relevantes encontradas no estado da arte serão apresentadas na seção dos trabalhos relacionados.

2.3. Considerações Finais

Esse Capítulo apresentou os principais conceitos relacionados às tarefas de processamento de vídeo, extração de características e de aprendizagem de máquina, contemplando a utilização de CNN e indutores rasos. Primeiramente aborda as tarefas de detecção de imagens e vídeos, destacando as suas peculiaridades, ou seja, a diferença de processar uma imagem e um vídeo. Posteriormente, enfatiza as técnicas de aprendizagem de máquina (dando ênfase a tarefa de classificação), onde são apresentadas as particularidades de vários classificadores rasos, e ainda se aprofundando nas arquiteturas

de aprendizagem profunda. Entre as arquiteturas de aprendizagem profunda os estudos foram direcionados a CNN, por ser mais adequada no processamento de imagens e *frames*. Em detalhes, para entender como funciona a arquitetura CNN, fundamentamos uma seção que estrutura a topologia CNN, e assim transcrevemos detalhes de como funcionam as principais camadas da CNN. Conhecer essa estrutura possibilita modificar as arquiteturas CNN ou ainda possibilita a criação de uma nova arquitetura. Por fim, encerra-se o Capítulo da fundamentação teórica. O próximo Capítulo apresenta os principais trabalhos relacionados a esta pesquisa.

Capítulo 3

Trabalhos Relacionados

A identificação do conteúdo no contexto em vídeos é uma tarefa desafiadora, e atualmente esse tema permanece aberto na literatura, de maneira especial quando retrata a detecção de conteúdo pornográfico, isso sem mencionar quando as abordagens são limitadas devido a restrições de hardware. Esse capítulo apresenta os trabalhos relacionados à pesquisa realizada, principalmente os relativos ao escopo e aos itens apresentados na fundamentação teórica.

3.1. Detecção de Conteúdo Pornográfico

Conforme disposto na literatura, existem diferentes propostas para detecção de conteúdo pornográfico, principalmente com foco em imagens. Yu e Han propõem uma abordagem de detecção de conteúdo pornográfico em imagens utilizando a segmentação de pele sobre o espaço de cor HSV (*Hue Saturation Value*). Os autores efetuam uma estimativa estatística sobre o percentual de pixels de pele associadas a operações morfológicas (Yu & Han, 2012). Nesse trabalho foi utilizado um *dataset* composto por 500 imagens advindas de filmes pornográficos e outras 500 imagens capturadas de vídeos de conteúdo não pornográfico. Observou-se uma limitação nesse trabalho, pois ao utilizar um *dataset* com uma quantidade pequena de amostras haverá dificuldade para classificar novas amostras.

Outra proposta baseada na segmentação de pele surge da combinação dos canais de diferentes espaços de cor (Rahman, et al., 2014). Inicialmente os autores convertem as imagens do espaço de cor RGB para HSV e YCbCr (*Luminance Chroma Blue and Chroma Red*). Para todo pixel de cada imagem é estabelecido um vetor de caracteristicas y = [H S Cb Cr], constituido pelo canal de matiz (H) e saturação (S) do espaço de cor HSV, além do canal de crominância azul (Cb) e vermelho (Cr) do modelo de cor YCbCr. Os autores dividem a imagem em blocos de tamanho 5x5, utilizando o cálculo da distância

de mahalobis é definido um limiar. Posteriormente são avaliados os pixels dos cantos e o pixel central de cada bloco, pixels correspondentes as coordenadas (1,1), (1,5), (3,3), (5,1) e (1,5). Na avaliação, se nenhum dos pixels do bloco nas coordenadas mencionadas for maior que o limiar definido, esse bloco será classificado como não contendo exposição de pele, ao contrário todo bloco é classificado como um componente de pele. Por fim é realizado uma estimativa estatística sobre o componente de pele, chegando a obter 93,9% na taxa de TP (*True Positive*) e 10,7% na taxa de FP (*False Negative*). Uma crítica ao trabalho é utilização de um *dataset* muito pequeno, sendo inferior a 100 amostras.

Macedo *et al*. propõem um método de detecção de pornografia infantil efetuando a combinação de características baseadas na detecção de face, idade e detecção de pornografia (Macedo, et al., 2018). Esse trabalho realizado em parceria com a Polícia Federal Brasileira possibilitou a criação de um novo *dataset* composto por 2138 imagens. Primeiramente os autores utilizaram o detector de face MTCNN, estruturado em uma arquitetura em cascata composto por três CNN consecutivas, este focado na tarefa da detecção da face. Na sequência, os autores utilizam uma CNN adaptada para tarefa da estimativa de idade para integrar o método de detecção de pornografia infantil, nessa etapa os autores utilizam a arquitetura VGG-16 aplicando *Transfer Learning* com os modelos treinados sob o *dataset* do Imagenet. Posteriormente foi utilizado uma rede para detecção de pornografia fornecido pelo Yahoo baseada na arquitetura Resnet. Os autores combinam as arquiteturas CNN destinadas a diferentes fins para fornecer uma nova ferramenta de detecção de pornografia infantil, possibilitando alcançar 79,84% em valores de acurácia.

Em (Steel, 2012), o autor propõe um classificador em cascata que efetua uma pré-filtragem das imagens, absorvendo características do tamanho da imagem e percentual de pixel correspondente ao componente de pele. O autor efetua a identificação da pornografia utilizando uma abordagem de *bag-of-visual-words* baseando-se no algoritmo de detecção de características SIFT (*Scale Invariant Feature Transform*). Em uma próxima etapa os autores utilizam o algoritmo Mask-SIFT a fim de obter um *corpus* de informações de histograma correspondente a imagens pornográficas e imagens não pornogáficas. Para realização dos experimentos, os autores utilizaram um *dataset* composto de três mil imagens adquiridas publicamente na *internet* composto por 1500 imagens pornográficas e 1500 imagens não pornogáficas. Por último, Steel estruturou em cascata os três métodos apresentados, chegando a obter uma taxa de 87% de TP sobre o *dataset* utilizado.

Ries e Lienhart fornecem um resumo contendo uma visão geral das abordagens de ponta no reconhecimento visual de imagens pornográficas, segmentando a pesquisa em três vertentes (Ries & Lienhart, 2004). Primeiro, discutem os trabalhos baseando-se em regiões de cor de pele, onde diferentes estratégias são descritas e construídas sobre o componente cor de pele. Um outro conjunto de trabalho foca na combinação de informações de forma associados a cor, abordagens baseadas em descritores de recursos locais. De forma similar, em (Karamizadeh & Arabsorkhi, 2018) os autores efetuam uma revisão dos principais métodos para detecção de pornografia em imagens.

Zhu *et al*. apresentam uma abordagem para a identificação de imagens com conteúdo pornográfico utilizando os componentes de pele, textura e geometria (Zhu, et al., 2007). Os autores estruturam seu modelo baseado na combinação do espaço de cor YIQ[5], YUV[6] e HSV. Para aprimorar o modelo, os autores aplicam um algoritmo de balanceamento branco para atingir as áreas de pele. Posteriormente, os autores adotam um modelo baseado em textura e geometria da estrutura humana, visando assim reduzir falhas advindas da região de fundo. As *features* tratadas nos modelos de cor, textura e geometria são extraídas e inseridas em um classificador SVM (*Support Vector Machines*) (Burges, 1998), permitindo classificar as imagens com contéudo pornográfico e não pornogáfico. O experimento foi realizado utilizando um *dataset* composto de 400 imagens pornográficas e 400 imagens não pornogáficas em contextos variados. Os autores reportam que o algoritmo proposto chegou a atingir 88,89% em valores de acurácia.

Outra estratégia é apresentada por (Deselaers, et al., 2008). Aqui os autores utilizam uma abordagem baseada em *bag-of-visual-words* para a identificação de imagens pornográficas. As imagens são representadas como um histograma de palavras visuais. As palavras visuais indicam *features* extraídas das imagens e um vocabulário é aprendido de maneira específica a partir da base de treinamento. As *features* locais são extraídas das amostras de imagens em torno de pontos de interesse com diferença gaussiana, dimensionadas para um tamanho comum e depois modificadas utilizando o procedimento estatístico PCA (*Principal Component Analysis*), deixando 30 coeficientes para reduzir sua dimensionalidade. Para criar um vocabulário visual foi utilizado um algoritmo de treinamento supervisionado de modelos de mistura gaussianos. Os autores utilizaram um *dataset* composto com 8500 imagens (aproximadamente 1700 imagens por classe)

[5] YIQ - Espaço de cores usado pelo sistema de TV em cores NTSC
[6] YUV - Espaço de cores usado pelo sistema de TV em cores PAL

estruturado em um conjunto de categorias. Foram estabelecidas regras para construção de um sistema de filtro baseado em um classificador que discrimina as cinco classes diferentes. Utilizaram um algoritmo SVM e LLM (*Lexical Level Matching*) sobre as *features* extraídas. Os autores reportam que o SVM foi consideravelmente melhor em relação ao LLM.

Em outro trabalho (Wiratama, et al., 2017), é proposto um método de detecção de objetos de pornografia utilizando o algoritmo Viola-Jones associado a algoritmos de detecção de pele realizado sob o espaço de cor YCrCb. Observando que algoritmo Viola-Jones é um método de detecção de objetos que pode processar rapidamente, os autores sugerem aplicações de uso em tempo real. Wiratama *et al.* estruturam a abordagem estabelecendo a Região de Interesse (ROI) do processo de detecção de objetos de pornografia. Primeiramente, o detector de objetos de pornografia Viola-Jones é utilizado e posteriormente o ROI processa um estágio de detecção de pele. Em seguida, implementam um aplicativo a fim de detectar pornografia. Este foi capaz de atingir 78,29% de acurácia com apenas 5% de FP em experimentos realizados sobre um *dataset* de 700 imagens, contendo 300 imagens pornográficas e 400 imagens não pornográficas.

Segundo exposto acima, diversos trabalhos utilizam propriedades do componente de pele para detecção do conteúdo pornográfico. Posteriormente tais abordagens são combinadas com outras estratégias. Por exemplo, estimativa estatística sobre as características de pele. Apesar de tais abordagens serem muito exploradas, acabam se tornando um modelo bastante limitado. Observando tal limitação, o presente trabalho propõe uma abordagem baseada na utilização de CNN, uma vez que atualmente a CNN se mostra como uma das abordagens mais promissoras em termos de acurácia. Nesse trabalho utilizamos a CNN como um extrator de características. Conforme já mencionado, as arquiteturas CNN são compostas por diferentes camadas, entretanto é necessário salientar que algumas dessas camadas são compostas por filtros. Referente a combinação dos canais de diferentes espaços de cor, foi definida a utilização de um único espaço de cor, por padrão adotado o espaço de cor RGB.

3.2. Detecção de Pornografia em Vídeos

Como disposto na literatura referente a detecção de conteúdo pornográficos em vídeos, uma das abordagens exploradas tem sido a análise de movimento, tal como disposto em (Endeshaw, et al., 2008) que extrai informações de movimentos pela análise

de *frames* adjacentes, gerando um vetor de movimentos para cada frame. Os experimentos foram realizados em um *dataset* composto de 400 arquivos de vídeo pornográficos e cerca de 350 arquivos de vídeos não pornogáficos inseridos em diferentes contextos. Experimentalmente, os autores examinaram a frequência do conteúdo e então definiram a região de janelas baseando-se na frequência. Posteriormente, avaliaram o desempenho da detecção de movimento repetitivo. Finalmente, efetuaram a detecção de pornografia em vídeos a partir das características de movimento baseadas na correlação temporal. Os autores utilizaram a representação de vetores de movimento como base para realizar uma estimativa espectral, possibilitando assim examinar a frequência da distribuição do movimento dominante em uma sequência de vídeo. Efetuando a extração da informação de movimento de dois *frames* consecutivos foram utilizadas a abordagem FBF (*First B-frame*) e a SBB (*Second B-frame*). Para estimativa espectral, cinco diferentes métodos foram empregados. Consequentemente foram obtidos resultados superiores 85% na taxa de TP, sendo inferior a 10% em relação taxa de FP na detecção de conteúdo pornográfico nos vídeos.

Outro estudo, focado na detecção de conteúdo pornográfico em vídeo foi proposto por Rea *et al*., surgindo um método a partir da combinação de características visuais associadas a características de áudio (Rea, et al., 2006). Entre as características visuais, os autores utilizam a detecção de regiões de pele e não pele obtidas por meio de histograma de referência. Para essa extração de *features* foi utilizado um projeto de filtragem *open-source* denominado Poseia. Os autores utilizam informações de cor sob as regiões de pele, pois essas ocupam uma faixa relativamente estreita no espectro de cores, desse modo fornecendo indícios de conteúdo pornográfico. Ainda entre as características visuais, foram utilizadas informações de movimento, advindas da extração dos vídeos MPEG construindo vetores de movimento. Um método de compensação global é aplicado sobre os vetores de movimento, em seguida efetuado um processo de filtragem e segmentação. Na sequência são utilizadas características de áudio, tais como métrica de periodicidade, energia do sinal e autocorreção no áudio. Os autores ressaltam que a combinação de diferentes características visuais em combinação com características de áudio são ideais para serem aplicadas em abordagens de detecção de conteúdo pornográfico em tempo real, principalmente quando disponibilizadas em vídeo *stream*.

Em (Lee, et al., 2009), os autores propuseram um sistema hierárquico multinível baseado em múltiplas características de cor, texto e forma. Características extraídas de diferentes domínios temporais, organizadas para determinar se o conteúdo do vídeo é

pornográfico. Para os experimentos, foi utilizado um *dataset* composto por 1000 vídeos contendo pornografia e 1000 vídeos normais de diferentes contextos. Após a extração dos *frames* dos vídeos utilizando um intervalo de 10 *frames* por segundo, foram obtidas um total de 630733 imagens. Os autores destacam a utilização do espaço de cor HSV como sendo um modelo de cor robusto quando aplicado no reconhecimento do componente de pele. Após a extração das *features*, um classificador SVM foi aplicado, possibilitando o desenvolvimento de um sistema para detecção de pornografia em tempo real. O sistema proposto utilizando o método hierárquico atingiu 94,2% na taxa de detecção e apenas 8% na taxa de FP.

Moreira *et al.* propõem análise do contexto de vídeos para detecção de violência e pornografia, utilizando características de áudio e conteúdo visual associadas a abordagens espaço-temporal para classificação dos vídeos (Moreira, et al., 2019). Nessa abordagem, efetuam a fusão multimodal para detecção de cena com conteúdo sensível. Os autores efetuam a combinação de diferentes classificadores destinados a tarefas de classificação específicas, tais como a de *frames* estáticos, fluxo de áudio e movimento de vídeo. Nesse trabalho, os autores utilizam o *dataset Pornography2k*, *dataset* composto de 1000 vídeos pornográficos e outros 1000 vídeos não pornográficos. Os autores destacam a eficiência na combinação de diferentes descritores de áudio e vídeo (PROS, MFCC[7], HOG[8] e TRoF[9]), chegando a obter 90,75% em termo de acurácia quando aplicado em um algoritmo THR (*Score Thresholding*).

Caetano *et al*. sugerem uma abordagem para a detecção de pornografia baseada na extração de características binárias locais e na representação de imagens, reforçando a utilização de descritores de vídeo aplicados a um classificador SVM (Caetano, et al., 2016). Nessa abordagem os autores utilizaram o descritor de vídeo BossaNova, relatando uma melhora em relação à precisão. As características extraídas foram aplicadas em um algoritmo SVM. Os experimentos foram realizados utilizando o *dataset Pornography-800*, composto por 400 vídeos pornográficos e 400 vídeos de conteúdo não pornográfico. Os autores reportaram uma taxa de 92,4% em termos de acurácia utilizando o descritor de video *bag-of-words (BoW-VD).*

Garcia *et al.* propõem a detecção de conteúdo pornográficos em imagens e vídeos, abordagem baseada na segmentação de pele em combinação com características dos

[7] MFCC – *Features* auditivas
[8] HOG – Descritor de imagem fixa
[9] TRoF – Descritor espaço temporal

pixels obtidos da aplicação de diversos filtros de textura sobre imagens/*frames* (Garcia, et al., 2018). Efetuando a avaliação em um *dataset* com 986 imagens e 253 vídeos, como resultado um filtro de conteúdo Web, que retorna à probabilidade da mídia (imagem/*frame*) conter pornografia. Efetuam a detecção e segmentação de regiões de pele utilizando o espaço de cor YCbCr e RGB em conjunto com características de textura da pele. Por fim, aplicando cálculos estatísticos, os autores reportam um acurácia de 80,23% sobre todo o *dataset*.

Retratando análise estatísticas, em (Polastro & Eleuterio, 2012) os autores efetuam a extração dos *frames* dos vídeos e submetem a uma ferramenta de detecção de nudez denominada NuDective que retorna o total de *frames* de nudez. Posteriormente, é aplicado uma análise estatística sobre os vídeos a fim de determinar automaticamente vídeos ilícitos. Os autores utilizaram um *dataset* composto de 150 vídeos. Nos experimentos os autores utilizaram quatro tamanhos diferentes de amostras (10, 20, 50 e 100 amostras), onde o melhor resultado foi obtido utilizando um tamanho 10 *frames*. Avaliando os experimentos os autores chegaram a obter uma taxa de 85,7% de TP e apenas 14,2% em relação a taxa de FP.

Dado os trabalhos dispostos no estado da arte relacionados com a detecção de conteúdo pornográficos em vídeos, observa-se que muitos dos trabalhos utilizam informação de movimento. Alguns autores efetuam a combinação de informação de movimento com outros componentes, tal como periodicidade no fluxo de áudio. Ainda, surgem abordagens que utilizam múltiplas características, efetuando uma fusão multimodal. Outra abordagem surge da extração das características do descritor BossaNova, aplicados posteriormente a um algoritmo SVM. Por fim, a utilização de filtros de textura em conjunto com ferramenta de detecção de nudez. Entretanto, todas as abordagens citadas acima possuem suas próprias limitações e a abordagem proposta neste trabalho acaba sendo estruturada de uma maneira singular. Com a adoção da CNN é possível ter um ganho significativo em relação à precisão do modelo quando comparado com as abordagens tradicionais que não utilizam CNN. Por fim, a abordagem aqui proposta é estruturada em uma janela de *frame* deslizante, tornando assim essa abordagem mais robusta, pois detectar contexto do vídeo por meio da avaliação de uma janela de *frames* é mais eficaz que a análise individual.

3.3. Detecção de Vídeo Utilizando Arquiteturas CNN

Incontáveis autores na literatura propuseram a utilização de CNNs voltadas especificamente para a classificação de vídeos. Comumente tais abordagens analisam o fator do movimento, que é uma importante propriedade para diferenciar um *frame* de outro. Existem diferentes propostas, tais como em (Karpathy, et al., 2014), onde os autores estendem as camadas convolucionais para trabalhar sob o domínio do tempo utilizando informações de espaço e tempo. Os autores utilizam o *dataset* Sports-1M que consiste em 200 mil vídeos e 4 milhões de *clips*, inseridos em diferentes contextos e estruturado em 487 classes. Os autores utilizam 70% dos vídeos para treinamento, 20% para validação e 10% para teste. Karpathy *et al.* utilizaram uma arquitetura CNN denominada *Slow Fusion* e a destacaram como uma rede bem representativa no reconhecimento do movimento. No treinamento ainda ressaltam a utilização de *Transfer Learning*. Os autores alcançaram 80,2% de acurácia avaliando por *hit top-5*, verificando se a classe correta consta no rank da probabilidade das cinco primeiras predições.

Li *et al.* propõem uma representação do espaço temporal multigranular para reconhecimento de ações realizada em vídeos, abordagem que utiliza as convoluções da CNN em diferentes planos (2D e 3D) associadas ao LSTM para modelagem temporal (Li, et al., 2016). Os experimentos foram realizados sobre os *dataset* HMDB51 e UCF101. O primeiro consiste em um *dataset* com 6849 vídeos estruturados em 51 classes, o segundo *dataset* composto por 13320 vídeos e organizado em 101 classes. Foi aplicada uma técnica de *Transfer Learning* utilizado um modelo treinado no *dataset* Sports-1M como entrada dos pesos no treinamento de um novo modelo. Entre as arquiteturas CNN utilizadas, os autores destacam o uso do VGG-19 e Alexnet. Para lidar com as *features* de movimento *stream,* os autores calcularam o fluxo ótico com a arquitetura VGG-19 e com a LSTM. Para lidar com *clips stream* os autores utilizam uma arquitetura C3D que é explorada para modelar videos clipes. Por fim, sobre a modelagem de video *stream*, os autores utilizam LSTM após a adoção da arquitetura C3D. Como resultado dos experimentos da fusão com MSD (*Multi-Granular Score Distribution*), os autores reportaram obter 64,1% de taxa de acurácia sobre o *dataset* HMDB51 e ainda 91.9% de taxa de acurácia sobre o *dataset* UCF101.

Também focadas na representação de nível de vídeo surgem outras abordagens da combinação CNN com o LSTM (Memória de Longo Prazo) (Ng, et al., 2015) (Wu, et al., 2015). Sendo o LSTM uma arquitetura de aprendizagem profunda que utiliza o

conceito de célula de memória (Jones, 2017), essa consegue manter um valor por um curto ou longo prazo dependendo da sua importância. Na literatura abordagens da combinação CNN com o LSTM utilizadas em tarefas voltadas para classificação de vídeos estão se tornando reincidentes, com indícios de que a célula de memória do LSTM seja capaz de armazenar informações de movimento.

Algumas abordagens atuais propõem resolver o problema de contexto, tal como o 3D CNN (Ji, et al., 2013) que extrai características das dimensões espaciais e temporais a fim de executar convoluções 3D, capturando informações de movimento codificadas em múltiplos quadros adjacentes. Tal abordagem trata o problema de contexto do vídeo utilizando as CNNs, entretanto surge outro problema: o custo computacional utilizado por essa estratégia. Considerando que, em geral, os dispositivos utilizados para gerar ou disseminar o conteúdo pornográfico possuem limitação de recurso computacional, a adoção de convoluções 3D CNN em dispositivos móveis (*smartphone* e *tablets*) tornasse impraticável. Para os experimentos os autores utilizaram dois *dataset* o TRECVID e o KTH. O *dataset* TRECVID 2008 consiste em 49 horas de vídeos capturados de 5 câmeras de vídeo localizados no aeroporto de Gatwick em Londres, focados no reconhecimento de três tipos classes, correspondente às ações (celular na orelha, pegar objeto e apontar). O *dataset* KTH consiste em um *dataset* estruturado em seis classes de ações. Os experimentos foram realizados utilizando as arquiteturas 3D CNN e HMAX, obtendo respectivamente 90,2% e 91,7% em termos de acurácia.

Em (Huang & Kong, 2016), os autores apresentam um CNN *ensemble* para classificar pornografia, conteúdo erótico e imagens não pornogáficas. Foi utilizado um *dataset* que consiste em 10997 imagens pornográficas, 1673 imagens com lingerie e 38832 amostras de imagens não pornogáficas. Os autores estruturam os experimentos utilizando um *ensemble* com sete arquiteturas CNN, e a partir da modificação da topologia da arquitetura CNN Alexnet foi obtido seis novas arquiteturas. Como resultado os autores reportam que a abordagem proposta supera em 2,93% uma abordagem com uma única arquitetura CNN, e ainda chegando a atingir 90,23% em termos de acurácia.

Já em (Huang & Ren, 2018) os autores relatam que as redes CNN tendem a causar *overfitting* no treinamento de dados quando utilizadas com pequena quantidade de dados. Para resolver isso, os autores apresentam um novo modelo de reconhecimento de imagens eróticas, adotam uma Rede Neural Convolucional Integrada ao *Bagging*, efetuando combinação de características de cor baseada em histograma com recursos de profundidade. Os autores utilizaram o *dataset* NPDI que consiste em 16727 amostras de

imagens pornográficas e não pornogáficas, foram reservadas 14727 imagens para treinamento e 2000 amostras para teste. Conforme os autores, considerando as vantagens e desvantagens do histograma de cores e das CNN, o modelo de rede neural convolucional integrado ao *Bagging* com recursos de cores permitem melhorar o desempenho dos métodos de reconhecimento de imagem erótica. Os autores relataram que a abordagem proposta atingiu 99,31% de acurácia sobre o *dataset* proposto, um incremento de 2,37% em relação a uma abordagem utilizando apenas CNN.

Em adição a classificação de imagens, Perez *et al*. apresentam o uso de CNN para classificação de vídeos pornográficos. A estratégia utilizada pelos autores incorpora uma nova abordagem por meio de combinação estática utilizando imagens e dinâmica, adicionando informação de movimento (Perez, et al., 2017). Os experimentos foram realizados em dois *datasets*. O primeiro o *dataset Pornography2k* que consiste em 1000 amostras de vídeos pornográficos e 1000 amostras de vídeos não pornográficos em diferentes contextos. O segundo *dataset* utilizado foi o *Pornography-800*, composto por 400 amostras de vídeos pornográficos e outras 400 amostras de vídeos não pornográficos. Os autores realizam os testes em três diferentes arquiteturas CNN, o GoogLenet, Alexnet e VGG. Referente à abordagem estática os autores aplicaram uma técnica de Transfer Learning, utilizando assim modelos pré-treinados com o Imagenet como pesos de entrada para o treinamento de um novo modelo. Sobre a abordagem de movimento, foram utilizadas informações do fluxo ótico em conjunto de vetores de movimento MPEG para treinar um novo modelo. Por fim, foi efetuada uma fusão entre *features* estáticas e de movimento, sendo esta a abordagem que obteve melhor resultado, atingindo 96,4% em termos de acurácia e 96,7% em relação ao F-Measure, sendo uma métrica mais adequada para avaliar a precisão do modelo observando um *dataset* desbalanceado.

Visando detectar conteúdo pornográfico em fluxos de vídeo, Ulges *et al*. efetuam uma combinação de diferentes técnicas de descrição de recursos, tais como histogramas de movimento, detecção de pele, palavras visuais e recursos de MFCC (*Mel Frequency Cepstral Coefficients*) baseados em áudio (Ulges, et al., 2012). Cada uma das *features* é fornecida a um classificador estatístico e então as pontuações da classificação resultantes são combinadas em uma única saída em uma etapa de fusão tardia. Para os experimentos os autores propõem um *dataset* a partir de um conjunto de 500 horas de vídeo obtidos do Youtube e dois *websites* pornográficos o RedTube[10] e PornHub[11] (sendo 1000 clipes

[10] https://www.redtube.com
[11] https://www.pornhub.com

pornográficos com uma média de 19 minutos e 2300 clipes do Youtube em média de 5 minutos). A abordagem da combinação de várias modalidades traz melhorias significativas na acurácia em comparação com o melhor resultado da abordagem unimodal. Essas melhorias foram percebidas em ambos os *dataset* no Redtube a taxa de FP reduziu de 10,5% para 6,8%, de forma similar para o PornHub a taxa de FP foi 9,8% para 4,3% nas amostras testadas.

Neste trabalho (Gangwar, et al., 2017), os autores propõem cinco diferentes métodos. Primeiramente um método de detecção de pele baseado na cor, nesta abordagem os autores trabalham sobre o espaço de cor YCbCr (Basilio, 2011), posteriormente aplicam estimativa estatísticas sobre o percentual de *pixels* de pele. O segundo método de detecção de nudez baseado na cor da pele, onde foi criado um modelo utilizando correlação e regressão linear baseado nos espaços de cor RGB, RGB Normalizado e HSV (Ap-apid, 2005). O terceiro método de detecção de pornografia utilizando um histograma sobre o espaço de cor HSV em conjunto com o algoritmo SIFT (Deselaers, et al., 2008) (Lopes, et al., 2009) e uma abordagem de agrupamento K-Means. O quarto método utilizando uma arquitetura CNN superficial semelhante ao Alexnet (Krizhevsky, et al., 2012), as arquiteturas foram treinadas utilizando *Transfer Learning* e foram usados os *frames* extraídos dos *dataset Pornography-800* e *Pornography2K*. Por fim, o quinto método estruturado sobre o *Open* NSFW, uma abordagem *open source* desenvolvida pelo Yahoo, sendo esta uma rede residual profunda de 50 camadas, treinada em imagens classificada como adequadas e não adequadas para os locais de trabalho (Kaiming, et al., 2015). Em seguida, os autores efetuam uma avaliação entre as diferentes abordagens em comparação aos trabalhos dispostos na literatura. Conforme os resultados expostos pelos autores, o quinto método (Open NSFW) foi o mais promissor e alcançou 87,56% em termos de acurácia.

Observando a complexidade e limitação das abordagens anteriores, é possível mencionar algumas propostas mais simples. Em (Zha, et al., 2015) os autores melhoram o desempenho da classificação realizando operações baseadas nas informações de movimento. Os autores utilizaram as arquiteturas CNN, Alexnet, Googlenet e VGG. Modificando a topologia da arquitetura CNN criaram sua própria arquitetura e experimentalmente várias configurações foram testadas. Os autores destacam que uma abordagem eficaz para a classificação de vídeo é extrair vários descritores de *features* de baixo nível e codificá-las em um vetor Fisher. Para os experimentos foram utilizados os *dataset* TRECVID MED'14 e o *dataset* UCF-101. O *dataset* TRECVID MED'14 consiste

em: (a) um conjunto de treinamento de 4992 vídeos com fundo não rotulados amostras negativas; (b) um conjunto de treinamento de 2.991 vídeos de amostras positivas (c) um conjunto de testes de 23953 vídeos que contém instâncias positivas e negativas dos eventos pré-especificados. A melhor abordagem proposta pelos autores foi a utilização de uma CNN estruturada com seis camadas ocultas combinadas com um vetor Fisher (FV) e trajetórias densas aprimoradas (IDT). Tal abordagem chegou a atingir 89,62% em termos de acurácia, sendo superior 1,02% em relação a melhor abordagem no estado da arte utilizando LSTM com imagem e fluxo ótico.

Nesse mesmo sentido, os autores (Jansohn, et al., 2009) efetuam combinação entre diferentes métodos convencionais aplicados em um classificador SVM, na sequência efetuam uma análise estatística sobre vetores de movimento. Para tal, primeiramente os autores efetuam a detecção de pele utilizando as seguintes *features*: mapa de probabilidade extraído do histograma do espaço de cor RGB, binarização definida por um limiar global sob determinadas regiões de pele, média da probabilidade de pele, taxa de *pixel* de pele e tamanho das regiões de pele. Essas *features* servem de entrada para classificação de um SVM. Uma segunda abordagem é a utilização do *bag-of-words* (BOVW) para obter informação de histograma local das imagens. Ainda efetuam a análise de movimento utilizando *features* de detecção de periodicidade (PER) por meio da identificação de padrões de movimento. Além disso, empregam a periodicidade de uma janela deslizante tendo em vista identificar movimentos repetitivos (PERWIN). Por fim, os autores destacam o uso de histograma de movimento (MHIST). Finalmente efetuam a fusão entre diferentes *features* para posterior classificação. Para os experimentos os autores utilizam um *dataset* composto por vídeos baixados do Youtube, o *dataset* consiste em 932 vídeos com conteúdo pornográficos e 2663 vídeos não pornográficos. A combinação entre as diferentes *features* melhora significativamente a redução da taxa de FP de 9,9% para 6,0%, conforme os resultados da combinação de palavras visuais e histogramas de movimento.

Abordando métodos baseados em multimodalidade, os autores propõem um método de detecção de vídeos adultos baseados em codificadores bimodais. Nessa abordagem, os autores utilizam análise de periodicidade dos segmentos de áudio em conjunto com a detecção de regiões de interesse baseados na análise de saliência (Liu, et al., 2017). Assim, a partir da combinação de ambas características foi gerado um método a fim de representar a semântica da ocorrência de sinais bimodais. Para os experimentos, os autores utilizam um *dataset* que consiste em vídeos adquiridos da *internet*, nos quais

48 vídeos pornográficos e 300 vídeos não pornogáficos foram separados para etapa de treinamento, e 50 vídeos pornográficos e 150 vídeos não pornogáficos foram reservados para teste. Liu et al. adotam *features* de movimento e cor como descritores globais e utilizam o SURF (*Speeded Up Robust Features*) como descritores locais. Posteriormente, aplicam um classificador SVM. Para o melhor caso, os autores reportam alcançar 96,7% na taxa de TP, enquanto apenas 10% em relação a taxa FP.

Conforme os trabalhos dispostos acima, atualmente a utilização de arquiteturas CNN para tratar o problema de detecção de imagens e vídeos são essenciais, tais abordagens mostram ser bastante efetivas. De maneira similar nosso trabalho também adota a utilização de CNN. Porém para classificação de um vídeo, utilizar apenas a saída da CNN não é suficiente e combinações adicionais acabam sendo necessárias. Contudo, tais abordagens suplementares incrementam a complexidade e muitas vezes inviabilizam à adoção em dispositivos com recursos limitados. Visando a tais limitações, a abordagem proposta foi estruturada de forma que a etapa secundária não necessite de tanto recurso computacional.

3.4. *Deep Learning* para *Mobile*

Relacionado a abordagens *Deep Learning* em dispositivos móveis, em (Chen, et al., 2016) os autores propõem um método para portar CNN em dispositivos móveis, em específico para plataforma iOS. A proposta visa a reutilização de dados para aliviar a carga nas camadas de convolução e ainda introduzem um método para remoção de *kernels* redundantes nas CNN, visando assim à redução do tamanho dos modelos.

Em (Sokolov & Patkin, 2018) é exposta uma arquitetura de rede neural leve para dispositivos móveis destinada a reconhecer as expressões faciais de pessoas capturadas pela câmera frontal. Os autores propõem uma aplicação multiplataforma voltada para o reconhecimento de emoções, projetada para uso em *smartphone* e *tablets*. Para tal, utilizam um modelo CNN treinado para identificar e reconhecer as emoções humanas em uma escala de excitação e valência. Os autores utilizam a arquitetura CNN Resnet, reportando uma taxa de 63,01% em termos de acurácia sobre um *dataset* de reconhecimento de face.

Em (Wang, et al., 2018), os autores fornecem uma visão geral dos desafios atuais e realizações que foram representativas sobre o avanço da aprendizagem profunda em

dispositivos móveis. O trabalho Wang *et al.* retrata três aspectos principais: (i) treinamento com dados móveis, (ii) inferência eficiente em dispositivos móveis e (iii) aplicativos de aprendizagem profunda móvel. Os autores propõem um método de aprendizagem profunda denominado DeepService, que processa informações do dispositivo tais como pressionamento de tecla e acelerômetro, visando identificar os usuários dos dispositivos móveis. No conjunto de teste não foi utilizado propriamente um *dataset* e os experimentos foram realizados diretamente com o usuário, a partir de um processo de experimentação.

Outra proposta utilizando *Deep Learning* em dispositivos móveis é apresentada em (Wu, et al., 2019). Neste trabalho, os autores expõem DeepShark, uma plataforma a fim de habilitar dispositivos móveis com a capacidade de alocação de recursos flexível no uso de sistemas de aprendizagem profunda comercial. Tal ferramenta foi projetada para balancear o tempo e a eficiência, aspectos relacionados ao espaço de memória e custo de energia.

Por outro lado, devido às limitações de memória dos dispositivos móveis, várias novas arquiteturas CNN foram propostas visando arquiteturas com modelos efetivamente menores (Anisimov & Khanova, 2017). Por exemplo, o Mobilenet (Howard, et al., 2017) proposto pelo Google, aplica funções convolucionais em profundidade e separáveis para diminuir significativamente o número de parâmetros usados, diminuindo assim as necessidades de memória. Da mesma forma, há outra arquitetura CNN chamada Squeezenet (Iandola, et al., 2016), que visa diminuir o consumo de memória e reduz o número de parâmetros necessários na CNN. Finalmente, o ShuffleNet (Zhang, et al., 2018) diminui seu tamanho de memória aplicando duas operações, ponto a ponto convolucional e troca de canais, onde o objetivo é reduzir as necessidades de memória e processamento.

Apesar de apresentar pouca necessidade de memória, a taxa de transferência de detecção em dispositivos móveis também foi almejada em alguns trabalhos relacionados. Por exemplo, a arquitetura YOLO (Redmon, et al., 2016) destinada a detectar objetos em imagem, sendo capaz de processar em tempo real, consegue atingir até 45 *frames*/seg. Da mesma forma, a arquitetura Fast R-CNN (Girshick, 2015) propõe uma abordagem de reconhecimento de objetos por meio da aplicação de redes convolucionais baseadas em região. Um aprimoramento é proposto pela arquitetura Faster R-CNN (Ren, et al., 2017), com limites previstos do objeto de acordo com sua posição na imagem.

Nossa proposta difere-se de outros trabalhos porque inicialmente utilizaremos um framework de Deep Learning denominado Caffe. Dado que o Caffe possui suporte a bibliotecas de kernel com aceleração computacional utilizando GPU e CPU, o que também possibilita a adoção das bibliotecas da Nvidia (CUDA). Esse framework foi selecionado devido à possibilidade de implementação utilizando a linguagem C/C++ (Andres, 2016), dada a necessidade de trabalhar em uma linguagem de baixo nível no Android. Ainda o que motiva nossa escolha é o suporte para o mobile Nvidia Tegra K1 GPU (Nvidia, 2019), o mesmo processador que será utilizado na placa de desenvolvimento da Nvidia Jetson Tk1. O kit de desenvolvimento Jetson TK1[12] será utilizado para simular a arquitetura de um dispositivo móvel. Dado o modelo CNN previamente treinado no Caffe, a abordagem proposta neste trabalho também foi estruturada sobre um dispositivo com recursos limitados, a fim de reproduzir a implementação em um cenário real. A fim de otimizar o processamento nos dispositivos móveis o presente trabalho propõe uma nova arquitetura CNN para obter assim um modelo a partir de uma arquitetura leve, projetada especificamente para dispositivos móveis, reduzindo a demanda computacional necessária mais principalmente preservando à acurácia.

3.5. Considerações Finais

Conforme observado, o presente capítulo foi organizado em quatros seções. Primeiramente uma seção destinada aos trabalhos no estado da arte que abordam a detecção de conteúdo pornográfico em imagens, e aqui constatou-se que muitos trabalhos utilizam a propriedade do componente de pele associado a estimativas estatísticas. Outras abordagens utilizam a combinação de *features* tais como cor, textura e geometria. Ainda são apresentados trabalhos estruturados a partir da combinação de diferentes canais, provenientes da utilização de vários espaços de cores distintos. Surgem também trabalhos utilizando descritores locais e globais em conjunto com a aplicação do modelo *bag-of-words*. Entre os classificadores utilizados, destaca-se o uso de SVM. Atualmente para detecção de imagens, as abordagens mais promissoras têm sido baseadas em abordagens que adotam a utilização de CNN. Um *review* dos autores e de abordagens utilizadas é

[12]https://developer.download.nvidia.com/embedded/jetson/TK1/docs/3_HWDesignDev/JTK1_DevKit_Specification.pdf

apresentado na Tabela 1, dispondo de informações das técnicas e algoritmos utilizados, estratégias adotadas, bem como o *dataset* e os espaços de cores.

Tabela 1 - Trabalhos relacionados sobre abordagens realizadas em imagens

Autores e Abordagens	RGB	YCrCv	HSV	YUV	YIQ	Detecção de Pele	Componente Textura / Geometria	Região de Interesse (ROI)	Segmentação de Pele	Estimativa Estatística	Bag of Words	Distância de Mahalobis	Operações Morfológicas	Combinação Features	Descritores Locais / Globais	Arquiteturas CNN	Algoritmo SIFT / Mask-SIFT	Viola-Jones	LLM - Lexical Level Matching	SVM	Transfer Learning	Imagenet	Proposto Inferior a 1000 amostras	Proposto Superio a 1000 amostras
[Yu & Han, 2012]	x	x				x			x	x			x										x	
[Rahman, et al., 2014]	x	x	x						x			x		x							x	x		
[Macedo et al.,2018]											x				x	x	x						x	
[Ries & Lienhart, 2004]						x			x					x	x									
[Karamizadeh & Arabsorkhi, 2018]						x			x	x				x	x									
[Zhu, et al., 2007]			x	x	x		x							x						x			x	
[Deselaers, et al., 2008]										x	x			x	x				x	x				x
[Wiratama, et al., 2017]		x						x										x					x	
	Modelo de Cor					Estratégias										Técnicas / Algoritmos						Dataset		

As duas seções subsequentes deste Capítulo, tratam de abordagens estruturadas em vídeos, primeiramente é apresentado os trabalhos relativos à detecção de pornografia em vídeos, e posteriormente os trabalhos pertinentes à detecção de vídeos adotando arquiteturas CNN. Em ambas as seções, foi observada uma certa tendência na utilização da informação de movimento, em geral combinadas com outras *features*. Atualmente tem destaque a utilização de arquiteturas CNN, muitas vezes combinadas com características de movimento. Alguns autores apresentam indícios que a combinação do uso de arquiteturas CNN em conjunto com a arquitetura LSTM gera resultados promissores. Observando a limitação dos dispositivos móveis, o presente trabalho está direcionado à utilização de apenas uma arquitetura de aprendizagem profunda, a arquitetura CNN. A Tabela 2 dispõe das principais abordagens para detecção de conteúdo em vídeos, onde são apresentadas diferentes estratégias, trazendo um apanhado geral das técnicas, algoritmos e *dataset* utilizados.

Tabela 2 - Trabalhos relacionados sobre abordagens realizadas em vídeo

Autores e Abordagens	Análise de Movimento	Vetor de Movimento / Fluxo Ótico	Correlação Espaço Temporal	Detecção de Regiões de Pele	Características de Áudio	Características Básicas (Cor, Forma, e Texto)	Descritores de Áudio e Vídeo	Informação Estática (Features de Imagens)	Espaço de Cor (RGB, YCrCb, HSV, outros)	Estimativa Estatistica	Estimativa Espectral	Histograma de Referência	Fusão Multimodal	Bag of Words	BossaNova	Bagging	LSTM	Combinação Features	Descritores Locais / Globais	Arquiteturas CNN	Algoritmo SIFT / Mask-SIFT	SVM	Transfer Learning / Imagenet	Pornography2k / Pornography-800	Sports-1M	HMDB51	TRECVID	NPDI	UCF101	Outros Dataset
[Endeshaw, et al., 2008]	x	x	x							x																				x
[Rea, et al., 2006]		x		x	x					x	x	x						x												x
[Lee, et al., 2009]			x	x		x			x													x								x
[Moreira, et al., 2018]		x			x		x	x					x					x						x						
[Caetano, et al., 2016]						x	x	x						x	x							x		x						
[Garcia, et al., 2018]				x				x	x									x												x
[Polastro & Eleuterio, 2012]				x		x			x	x																				
[Karpathy, et al., 2014]	x	x																		x			x		x					
[Li, et al., 2016]	x	x	x														x			x			x		x	x			x	
[Ng, et al., 2015]	x	x															x	x		x										x
[Wu, et al., 2015]	x	x															x	x		x										x
[Jones, 2017]	x	x															x	x		x										x
[Ji, et al., 2013]	x	x	x																	x			x				x			
[Huang & Kong, 2016]								x										x		x										x
[Huang & Ren, 2018]		x		x					x			x				x		x		x								x		x
[Perez, et al., 2017]	x	x	x			x	x	x					x					x		x			x	x						
[Ulges, et al., 2012]	x	x	x	x	x	x	x			x										x										x
[Gangwar, et al., 2017]	x	x		x		x		x												x	x		x	x						
[Zha, et al., 2015]	x	x			x		x											x		x							x		x	
[Jansohn, et al., 2009]		x			x							x		x					x	x		x								x
[Liu, et al., 2017]	x	x		x	x		x						x					x	x	x		x								x
	Features									Estratégias										Técnicas				Dataset						

Por fim, a última seção dos trabalhos relacionados transcreve as abordagens de aprendizagem profunda destinada a dispositivos móveis. Sobretudo essa seção destaca alguns fatores importantes a serem analisados para o uso e implementação de abordagens de aprendizagem profunda em dispositivos com recursos limitados. Assim, com objetivo de otimizar o processamento nos dispositivos móveis uma nova arquitetura CNN tende a ser criada: um modelo a partir de uma arquitetura leve, projetada especificamente para dispositivos móveis, reduzindo a demanda computacional necessária, mas principalmente preservando à acurácia.

Desse modo, efetuamos uma síntese dos trabalhos relacionados com este projeto, assim finalizando o capítulo de trabalhos relacionados. Dando sequência ao trabalho, o Capítulo 4 apresenta os procedimentos metodológicos desta dissertação.

Capítulo 4

Procedimentos Metodológicos

Este capítulo propõe um novo método para detecção de pornografia em vídeo, denominado *Stacking Meta Learner*[13], abordagem estruturada duplamente e sensível ao contexto. Posteriormente ressalta a utilização de três métodos para otimizar a abordagem proposta: a amostragem de *frames*, uma CNN leve em tempo real e um verificador de confiança da CNN.

4.1. Detecção de Pornografia Baseado em Sequência de *Frames*

Dado o problema exposto, este trabalho propõe uma nova abordagem para detecção de vídeos com conteúdo impróprio em uma granularidade de *frames*. Primeiramente é efetuada a classificação dos *frames* utilizando um modelo CNN previamente treinado. Em sequência é utilizada a saída da CNN como entrada para um extrator de características baseado em uma janela de *frame* deslizante. As características absorvidas de toda janela são aplicadas em um algoritmo de reconhecimento de padrões. Uma visão geral do *Stacking Meta Learner* pode ser visualizada na Figura 8.

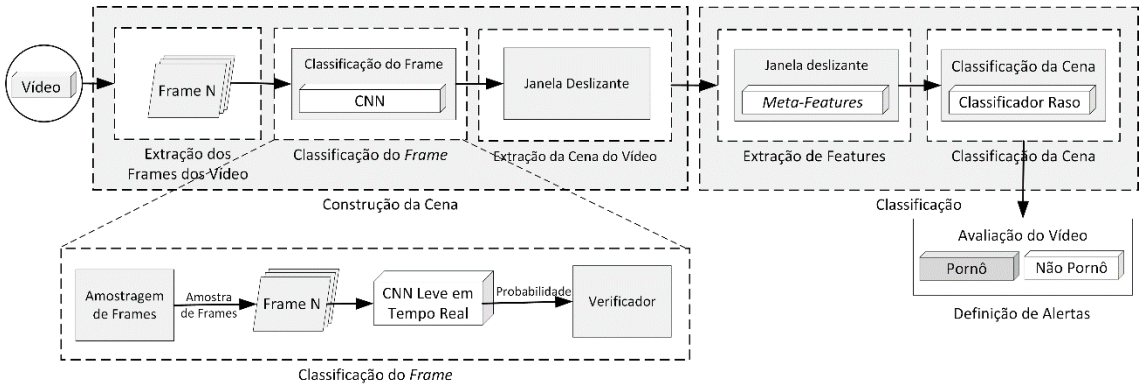

Figura 8 - Stacking Meta Learner - Abordagem de detecção de conteúdo impróprio sensível ao contexto.

[13] O termo *Meta Learning* também é utilizado para referenciar a esquemas de aprendizagem ensemble (Seewald, 2002). Essa abordagem parte de um tipo específico de *Meta learning* denominado *Stacking* que constrói os modelos em ensemble a partir de diferentes algoritmos de aprendizagem (Enesco & Dhandhania, 2018) (Seewald, 2003).

Inicialmente foram utilizadas três arquiteturas CNN, o Caffenet, Alexnet e Googlenet. As arquiteturas foram usadas preservando sua topologia padrão, primeiramente treinadas de maneira convencional e posteriormente empregando uma técnica de *Transfer Learning*. Foram gerados seis modelos CNN, sendo que para cada um dos modelos foi implementado a abordagem *Stacking Meta Learner*. A abordagem *Stacking Meta Learner* foi estruturada em dois estágios principais: *construção* e *classificação de cenas*.

A *construção da cena* é realizada pela extração dos *frames* pornográficos do vídeo e pela construção de uma janela deslizante de amostras de *frames*. Entre a etapa de extração dos *frames* dos vídeos e a extração da cena, são aplicados três métodos de otimização de recursos: *Amostragem de frames*, *CNN leve em tempo real* e módulo *verificador*. Os métodos de otimização foram estruturados para reduzir a demanda computacional, direcionados a dispositivos com restrições de recursos hardware. Adiante das otimizações realizadas no módulo de classificação, a janela deslizante de *frames* é estruturada. A janela deslizante compreende um conjunto de *frames* adjacentes ao *frame* avaliado, nomeado como *frame* pivô. Em outras palavras, cada janela deslizante contém amostras dos *frames* dos vídeos, segundos antes e depois de ocorrer um potencial *frame* pornográfico no *frame* pivô. O tamanho da janela foi definido experimentalmente, utilizando diferentes tamanhos de janela até obter a melhor configuração.

A proposta do módulo de classificação de cenas envolve a extração da *features* e classificação de cena utilizando as *meta-features*, conforme disposto na Figura 9. O *extrator de características* recebe como entrada os valores de confiança da CNN da classe pornográfica do *frame* pivô e *frames* adjacentes, e então constrói um conjunto de *meta-features* em relação ao contexto da cena. (Figura 9, *Extrator de características*). Consequentemente, na classificação aplica-se um *classificador raso* sobre o conjunto de *meta-features* obtido, determinando se a cena contém conteúdo pornográfico ou não (Figura 9, *Classificador raso*).

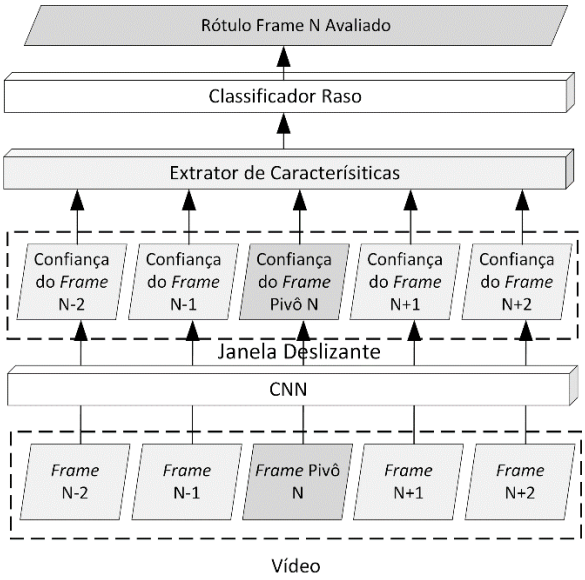

Figura 9 - Classificação de *frame* de acordo com janela deslizante de cena

Para criar o conjunto de *meta-features*, o extrator de características analisa todos os *frames* da cena. Dessa maneira, o *extrator de características* recebe como entrada os valores de confiança da CNN para cada *frame* da cena. Os *frames* classificados pela CNN são então usados para criar um conjunto de *meta-features* contendo informações sobre o contexto da cena em que o *frame* pivô é inserido. O conjunto de *meta-features* extraídas compreende em:

a) ***Frame Base:*** Valor do *frame* que está localizado na posição central da janela de *frame,* podendo também ser representado como sendo um pivô. Esse atributo corresponde a uma tradução direta da saída da CNN. Consequentemente, representa o rótulo atribuído ao *frame* analisado. Definido S como sendo o tamanho total da janela de *frames*, F_b é o *frame* pivô representado pela Equação 2:

$$F_b = índice \quad \text{ou} \quad \text{notação adicional: } F_b = \frac{S}{2} + 1 \tag{2}$$

b) ***Intervalo Percentual:*** Essa *meta-feature* indica a proporção de amostras de *frames* classificados como pornográficos, definindo o percentual de incidências de *frames* pornográficos na decorrência do tamanho da janela estabelecida. Dessa maneira, é possível avaliar as amostras de *frames* classificados como pornográficos na cena de acordo com a saída da CNN. Sendo $P_{internal}$ o percentual do tamanho total da janela e x_i *frame* na posição corrente, segue a representação pela Equação 3:

$$P_{internal} = \frac{\sum_{i=0}^{S} f(x) x_{i=porn}}{S} \quad (3)$$

c) ***Percentual Anterior*:** Essa *meta-feature* compreende a proporção de amostras de *frames* com classificação pornográfica ocorridas antes do *frame* pivô na cena especificada. Em outras palavras, mede-se a taxa de ocorrência de amostras de *frames* com classificação pornográfica antes do *frame* pivô. Sendo $P_{previous}$ o percentual de *frames* pornográficos da janela anterior de *frames*, segue exposto a Equação 4:

$$P_{previous} = \frac{\sum_{i=0}^{F_b} f(x) x_{i=porn}}{\frac{S}{2}} \quad (4)$$

d) ***Percentual Posterior:*** Essa *meta-feature* compreende a proporção de amostras de *frames* classificados como pornográficos ocorridos após o *frame* pivô na cena especificada. Isso significa que são medidas se mais amostras de *frames* com conteúdo com classificação pornográfica ocorrerão após o *frame* pivô. Sendo P_{Next} o percentual de *frames* pornográficos da janela posterior de *frames*, segue a Equação 5:

$$P_{Next} = \frac{\sum_{i=F_b}^{S} f(x) x_{i=porn}}{\frac{S}{2}} \quad (5)$$

e) ***Período Anterior:*** Essa *meta-feature* compreende o intervalo de amostras de *frames* antes do *frame* pivô, no qual ocorreu outro *frame* com classificação pornográfica. Assim, indicando se outro *frame* classificado como pornográfico ocorreu em um curto período. Partindo do *frame* pivô e retrocedendo a janela, é o número de posições deslocadas do *frame* pivô até localizar o próximo frame pornográfico. Não sendo localizado nenhum *frame* pornográfico na extensão da janela anterior, será fixado o valor máximo das posições percorridas. Sendo $T_{previous}$ a quantidade de posições de *frames* deslocados do *frame* pivô retrocedendo até o próximo *frame* pornográfico, e Q_{frame} correspondente posições de deslocamento, conforme exposto na Equação 6:

$Enquanto\ x_i \neq porno\ faça$:

$$T_{previous} = \sum_{i=F_b}^{F_b > 0} Q_{frame} + 1 \quad (6)$$

f) ***Período Posterior:*** Essa *meta-feature* compreende o intervalo de *frames* após o *frame* pivô, no qual ocorreu outra amostra de *frame* com classificação pornográfica. Assim, indicando se ocorrerão mais *frames* pornográficos após o *frame* pivô. Partindo do *frame* pivô e avançando as posições da janela, correspondem ao valor da quantidade de posições deslocadas do *frame* pivô até localizar o próximo *frame* pornográfico. Não sendo localizado nenhum *frame* pornográfico na extensão do tamanho da janela posterior, será fixado o valor máximo de posições percorridas. Sendo T_{next} a quantidade de posições de *frames* deslocados do *frame* pivô avançando até o próximo *frame* pornográfico, segundo exposto na Equação 7:

$Enquanto\ x_i \neq porno\ faça$:

$$T_{next} = \sum_{i=F_b}^{S} Q_{frame} + 1 \quad (7)$$

Assim, o conjunto de *meta-features* obtido compreende em 6 características, extraídas pela análise do contexto da cena. Finalmente, baseado nas *meta-features* obtidas o *frame* pivô é classificado utilizando um classificador raso. Consequentemente, a classificação da cena recebe com entrada a classe atribuída ao *frame* pivô, construindo um conjunto de *meta-feature* e reclassificando o *frame*-pivô, levando em consideração toda a janela deslizante.

Em regras gerais, para estabelecer se um *frame* pivô contém contexto pornográfico ou não avaliando se uma única amostra é classificada como pornográfica em uma janela de tempo, indica com alta probabilidade um falso positivo. Por outro lado, se várias amostras de *frames* adjacentes são classificadas como pornográficas, a probabilidade de o contexto pornográfico ser um verdadeiro positivo é maior.

Consequentemente, o classificador é capaz de detectar conteúdo pornográfico usando características baseadas no contexto, melhorando assim a classificação final.

4.2. Métodos para Otimização de Recursos

Visando a otimização da abordagem *Stacking Meta Learner* na detecção de conteúdo de vídeo em tempo real em dispositivos com recursos limitados, são propostas três técnicas: *amostragem de frames, CNN leve em tempo real* e um *verificador*, como

mostra a Figura 10. O objetivo é aumentar o rendimento da detecção, diminuir as demandas de memória e manter ou até mesmo melhorar a precisão da detecção.

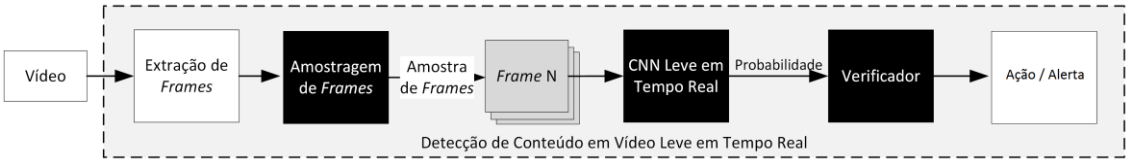

Figura 10 - Proposta de otimização para o *Stacking Meta Learner*, arquitetura de detecção de conteúdo de vídeo em tempo real para o processamento de dispositivos restritos.

A técnica de *amostragem de frames* visa diminuir o número de *frames* que serão classificados. Para esse fim, a abordagem proposta mostra os *frames* de vídeo conforme eles ocorrem, diminuindo consequentemente as demandas de processamento. Em seguida, a *CNN leve em tempo real* classifica os *frames* selecionados. A *CNN leve em tempo real*, por sua vez, é construída levando em consideração a acurácia, a memória e as taxas de transferência de detecção. Para atingir esse objetivo, nossa proposta combina a arquitetura da CNN de acordo com as demandas de processamento e memória de camadas, além de levar em consideração a precisão obtida. Finalmente, o módulo *verificador* visa melhorar a acurácia do sistema sem demandas adicionais de processamento. Para esse fim, aplica-se uma verificação na saída da classificação de acordo com os valores de confiança da CNN.

As próximas subseções descrevem em detalhes esses três módulos, incluindo a fase de construção da CNN proposta e o módulo *verificador*.

4.2.1. Amostragem de Frames

Como descrito anteriormente, um vídeo é feito por uma sequência de *frames* (Seção 2.1.2), por exemplo, 23,97 *frames*/seg. Em geral, as abordagens de detecção de conteúdo de vídeo na literatura avaliam todos os *frames*, independentemente dos requisitos do aplicativo. Por exemplo, um aplicativo de *streaming* de vídeo não requer a avaliação de todos os *frames* do vídeo, ao contrário, apenas um subconjunto de *frames* do vídeo pode ser avaliado, como aqueles nos quais o conteúdo do *frame* muda significativamente, reduzindo assim as demandas de processamento.

Diante disso, o módulo de *amostragem de frames* seleciona os *frames* de vídeo que serão avaliados. Consequentemente, as demandas de processamento podem ser significativamente diminuídas, pois nem todos os *frames* precisam ser avaliados pela

abordagem de classificação *Stacking Meta Learner*, como no módulo *amostragem de frames* apresentado em destaque na Figura 11. Para selecionar os *frames* de vídeo, a amostragem de *frames* pode ser implementada de maneira proativa ou reativa. Proativamente, os *frames* de vídeo são selecionados de acordo com a alteração do conteúdo. Por exemplo, eles podem ser avaliados apenas em relação aos *frame-chave* de vídeos codificados em MPEG (Perez, et al., 2017). Por outro lado, a abordagem reativa seleciona os *frames* de vídeo, independentemente do seu conteúdo. Por exemplo, selecione e avalie após cada N *frame* de vídeo. Portanto, cada técnica de seleção de *frames* deve ser selecionada de acordo com as demandas da aplicação e é específica ao contexto.

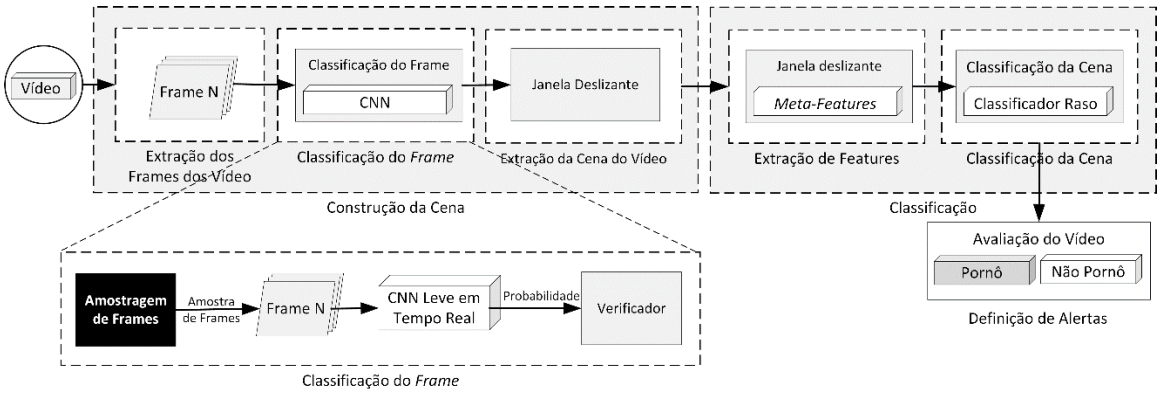

Figura 11 - Módulo *amostragem de frames*

A *amostragem de frames* proposta foi implementada de forma mista. Primeiramente de forma reativa, ao ínves de utilizar todos os *frames* do vídeo foi estabelecida uma variável denominada *amostragemFrames* que define a periodicidade em que os *frames* são capturados, e um contador é incrementado à medida que os *frames* são descartados, o que determina o intervalo de amostragem. Assim que o valor do contador se iguala a varíavel "*amostragemFrames*", o frame é capturado e armazenado na janela de *frames* e, por fim, o contador é zerado. De maneira proativa o módulo de amostragem analisa a janela de *frames*, e quando identificado um frame pornográfico, o módulo de amostragem é temporariamente desabilitado (para variável "*amostragemFrames*" é atribuido o valor zero), para que toda a sequência de frame seja analisada pelo *Stacking Meta Learner*. O módulo de amostragem é novamente habilitado quando em toda a extensão da janela de *frames* não é encontrado nenhum frame pornográfico (a variável "*amostragemFrames*" recebe o valor do intervalo entre os *frames*).

Portanto, a abordagem de *amostragem de frames* pode diminuir significativamente as demandas de processamento. Contudo, introduz um atraso de

detecção e possíveis perdas de *frames*. Anteriormente, o usuário só seria alertado após um período, referente à ocorrência anterior do *frame*. Por exemplo, após a passagem de *frames* N-1. Adiante, a técnica de detecção pode perder alguns *frames* de vídeo, não alertando o usuário. No entanto, é importante observar, em geral a detecção de conteúdo de vídeo procura por conteúdo que ocorre em vários *frames*. Tal como ocorre na detecção de conteúdo pornográfico em vídeos. Nesse caso, esse tipo de *frame* ocorre em vários *frames* sequenciais, em vez de ocorrer em um único *frame*. Portanto, a abordagem proposta pode aproveitar adequadamente a técnica de *amostragem de frames*.

4.2.2. CNN Leve em Tempo Real

Em geral, as arquiteturas CNN são avaliadas com relação às medidas de acurácia, memória ou taxa de transferência (*throughput*). Portanto, em geral, os autores costumam realizar modificações na arquitetura da CNN para melhorar uma medida específica, e.g., acurácia, enquanto afeta outras medidas, e.g., memória. No entanto, para a implantação adequada de dispositivos com recursos limitados, as arquiteturas CNN devem fornecer alta precisão, baixo nível de memória e alta taxa de transferência na detecção. Portanto, a precisão, a memória e o rendimento da detecção devem ser avaliados juntos na fase de implementação da CNN, na abordagem que confere a etapa em destaque na Figura 12. Tendo como resultado uma arquitetura CNN otimizada, a disposição das camadas deve ser avaliada de acordo com a disponibilidade e os requisitos dos recursos do dispositivo de destino. Consequentemente, a arquitetura CNN construída poderá fornecer uma *CNN leve em tempo real*, apropriada para a implantação de dispositivos com recursos limitados.

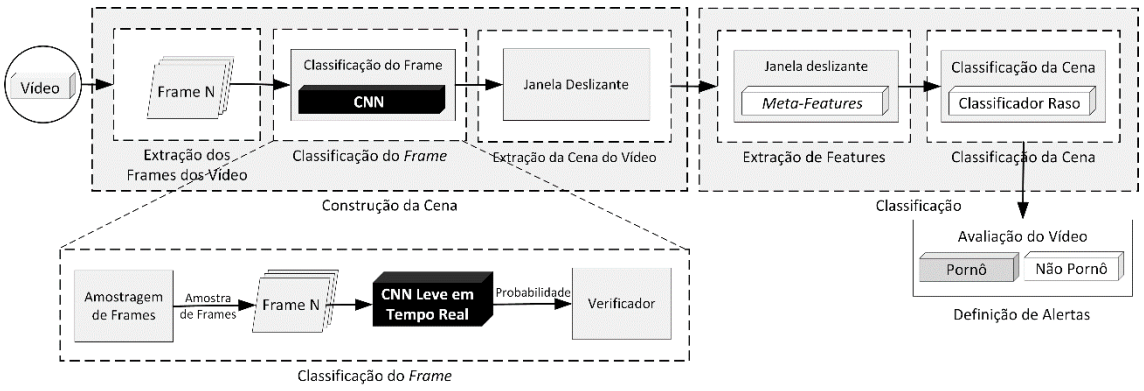

Figura 12 - Módulo CNN leve em tempo real

Nesse sentido, uma nova métrica para avaliar o desempenho da CNN é proposta a leveza. A *métrica de leveza* é calculada com a soma dos valores normalizados de memória, taxa de transferência de detecção e acurácia. O processo de cálculo leva em consideração o conjunto de arquiteturas da CNN avaliadas e normaliza todos os valores obtidos, usando o processo de normalização Z-Score (Jain, et al., 2005). Assim, para o cálculo do *Z-Score* foi adotado a Equação 8:

$$z = \frac{(X - \mu)}{\sigma} \tag{8}$$

Consequentemente, cada medida de desempenho tem o mesmo peso no cálculo da *métrica de leveza*. O cálculo da métrica de leveza é obtido da soma da normalização *Z-score* para cada uma das arquiteturas CNN modificadas, individualmente calculadas para cada uma das variáveis: acurácia, *throughput* e memória. A *métrica de leveza* foi expressa na Equação 9, definida como *"lw"*, nesta equação *"n"* representa a quantidade de arquiteturas avaliadas e *"i"* o índice das variáveis a serem normalizadas. Na normalização, *x*, *y* e *z* correspondem ao valor respectivamente de acurácia, *throughput* e memória. Sendo x_i, y_i e z_i o valor de cada variável, μ_x, μ_y e μ_z o cálculo da média, e por fim, σ_x , σ_y e σ_z é o desvio padrão das suas respectivas amostras.

$$lw = \sum_{i=0}^{n} \frac{x_i - \mu_x}{\sigma_x} + \frac{y_i - \mu_y}{\sigma_y} + \frac{z_i - \mu_z}{\sigma_z} \tag{9}$$

Como resultado, a medida leve inclui um conjunto de propriedades desejadas na CNN para a implementação adequada em dispositivos com recursos limitados. Portanto, a proposta *métrica de leveza* permite avaliar devidamente as arquiteturas CNN para detecção da CNN em tempo real. Para tal, foi efetuado a modificação da topologia da arquitetura tradicional do Caffenet, adicionando e removendo os grupos de camadas, permitindo utilizar a *métrica de leveza* na avaliação das diferentes disposições da arquitetura CNN.

4.2.3. Verificador de Confiança da Saída da CNN

Em geral, para alcançar baixas demandas de memória e maior taxa de transferência de detecção, as arquiteturas CNN são modificadas e uma troca na acurácia

é evidenciada. Contudo, degradações na acurácia pode tornar alguns aplicativos inviáveis para implantação, até em dispositivos com recursos limitados. Consequentemente, apesar de diminuir as demandas de memória e aumentar a taxa de transferência de detecção, as taxas de acurácia devem permanecer semelhantes, ou até mesmo superiores em relação aos resultados obtidos com a disposição padrão das camadas da CNN. Por outro lado, para aumentar a acurácia da detecção, em geral, é preciso executar mais processamento de dados, diminuindo o rendimento da detecção.

Diante disso, foi proposto um módulo *verificador* que verifica os valores de confiança na saída da CNN. O módulo *verificador* aceita ou não uma classificação CNN de acordo com o valor de confiança da saída da CNN, módulo verificador situado na abordagem em destaque disposto na Figura 13. Para esse fim, o módulo *verificador* verifica se o valor de confiança é maior que um determinado valor limiar. Consequentemente, apenas classificações de alta confiança são aceitas, aumentando assim a acurácia do sistema de maneira leve. Isso porque para cada classificação apenas uma verificação simples no limiar deve ser realizada, sem a necessidade de qualquer cálculo adicional (ou modificação da CNN).

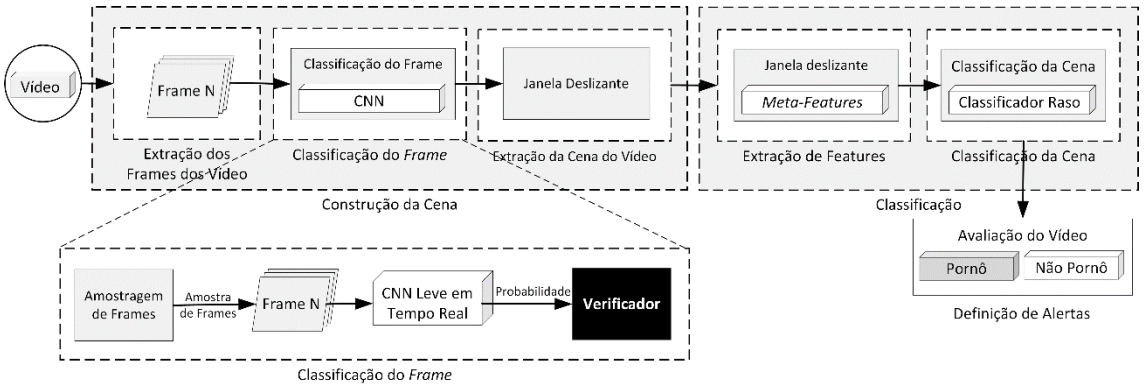

Figura 13 - Módulo *verificador*

Portanto, ao aplicar o módulo *verificador*, o usuário pode garantir que apenas os *frames* altamente confiáveis sejam aceitos e classificados. Como resultado, alarmes falsos podem ser reduzidos enquanto as demandas de processamento permanecem baixas.

4.3. Considerações Finais

O *Stacking Meta Learner* sendo uma abordagem de detecção de conteúdo pornográfico baseado em uma sequência de *frames*, torna-se sensível na identificação do

contexto dos vídeos. Assim, dessa forma os *frames* são analisados em conjunto, não sendo apenas classificados individualmente. A presente abordagem foi estruturada na sequinte suposição, a classificação do contexto do vídeo por meio da avaliação da janela de *frames* é mais eficaz que a análise individual utilizando apenas a saída tradicional de uma CNN.

Uma possível limitação associada a esta abordagem é determinar uma janela de *frames* que represente bem as características do contexto do vídeo, pois, ao definir uma janela muito pequena, não é possível extrair as características suficientes para percepção do contexto do vídeo. Em contrapartida, determinar uma janela muito grande de *frames* tende a não ser uma boa abordagem, observando a inserção dos *frames* delimitadores no início e no fim do vetor de características, incluídos na etapa de extração do *Stacking Meta Learner*.

As características preliminares selecionadas para o classificador *Stacking Meta Learner* demonstraram eficiência. Entretanto, ainda existe a possibilidade de adicionar ou remover características a fim de otimizar o modelo. Consideramos que características relativas ao movimento sejam interessantes para experimentos futuros, conforme observado em experimentos correntes.

A proposta da abordagem *Stacking Meta Learner* monstrou-se promissora, possibilitando a detecção do conteúdo pornográfico em vídeos por meio de uma janela deslizante de *frames*. Essa abordagem foi estruturada de maneira dupla, utilizando um modelo CNN previamente treinado e um indutor para classificação das instâncias da janela de *frames*. Tal abordagem foi projetada para abstrair a complexidade, visando a implementação em dispositivos com recursos limitados.

Outro fator motivador dessa proposta é a utilização da abordagem *Stacking Meta Learner* para aplicações em tempo real. Os *frames* são analisados praticamente em tempo real tendo apenas um *delay* referente ao tamanho da janela de *frames*. Desse modo, não é necessário obter todos *frames* do vídeo para posteriormente classificá-los, ou seja, a classificação em tempo real é realizada na janela de *frames* à medida que ela vem sendo preenchida (alimentada).

Assim a abordagem *Stacking Meta Learner* é capaz de identificar trechos pornográficos contidos nos vídeos em um espaço de tempo no melhor caso de 4 segundos, cálculo baseado no tempo necessário para preencher a janela deslizante, do primeiro *frame* até o *frame* pivô. Ou seja, levando em consideração o tamanho da janela (10 *frames*

anteriores mais o *frame* pivô*)* e a aquisição de um a cada 10 *frames* em uma taxa de 23,97 FPS.

No pior caso, serão necessários 7 segundos para determinar se o trecho do vídeo em análise possui caráter pornográfico ou não. Tempo necessário do deslocamento dos *frames* do vídeo para o preenchimento da janela, onde foi verificado um impasse na janela anterior (dado 10 *frames* anteriores ao pivô, 5 *frames* não pornográficos e outros 5 *frames* pornográficos), serão necessários no mínimo a análise de mais 5 *frames* para determinar sua classe correspondente. Ou seja, para determinar a classe do *frame* em questão, é necessário que a metade da janela mais um *frame* correspondam a mesma classe.

A arquitetura de detecção de conteúdo de vídeo em tempo real destinada a dispositivos com recursos limitados aproveita as configurações do aplicativo nas quais normalmente é empregada uma técnica de detecção de conteúdo de vídeo. Em geral, essas abordagens são usadas em aplicativos em tempo real, tal como as plataformas de *streaming* de vídeo. Consequentemente, é efetuado um aperfeiçoamento no *Stacking Meta Learner*, a fim fornecer a detecção em tempo real de maneira otimizada de forma tríplice. Primeiro, a amostragem de *frames* seleciona apenas um subconjunto dos *frames* de vídeo para avaliação. Essa abordagem reduz significativamente as demandas de processamento, pois nem todos os *frames* dos vídeos são avaliados. Segundo, para fornecer uma *CNN leve em tempo real*, propomos uma nova técnica de medida de desempenho, a métrica "*leveza"*. Tal medida, leva em consideração a memória, a taxa de transferência de detecção e a acurácia, que são os parâmetros que foram definidos para selecionar a melhor arquitetura CNN para implementação do modelo. Finalmente, o módulo *verificador* aceita apenas uma classificação altamente confiável, aumentando a acurácia do sistema com pouco ou nenhum impacto no desempenho. Portanto, a abordagem proposta, diferentemente dos trabalhos relacionados, aborda a detecção de conteúdo de vídeo em tempo real, levando em consideração a natureza da restrição de recursos dos dispositivos nos quais serão implantados.

Capítulo 5

Análise e Resultados

Este capítulo apresenta os resultados obtidos nesta dissertação.

5.1. *Dataset* de Vídeo Pornográfico

Atualmente não existe um *dataset* que seja referência para detecção de conteúdo pornográfico em vídeos. Uma proposta apresentada por (Perez, et al., 2017) fornece o *Pornography-2k*, um *dataset* composto por arquivos em formato de vídeo, ou seja, esse *dataset* apenas rotula os vídeos como pornográfico e não pornográfico, os *frames* não foram extraídos e rotulados individualmente. Essa é uma forte limitação, pois um único vídeo pode ser criado por ambos os tipos de categorias, já que, em geral, nem todos os *frames* extraídos de vídeos pornográficos possuem teor pornográfico. Portanto, para avaliar adequadamente as técnicas de classificação projetadas para esses fins, é preciso primeiro construir um conjunto de dados adequado. Assim foi observada a necessidade da criação de um novo *dataset*. Para isso, os *frames* dos vídeos foram extraídos e rotulados manualmente.

Para criação do novo *dataset* foi efetuada a extração dos *frames* de um total de 14671 vídeos, obtidos nos formatos .mp4, .3gp e .avi. Esses vídeos são compostos por pessoas de diferentes etnias (branco, negro e asiático), inseridas em diferentes contextos. Para composição dos vídeos não pornográficos utilizamos o *dataset* UCF101 (Soomro, et al., 2012). Para classe de vídeos pornográficos, o material foi baixado de diferentes sites pornográficos de domínio público. Efetuando uma análise em relação a extração dos *frames*, foi observada a transição de cena e de movimento, observando a mudança significativa de um *frame* para outro, após avaliar diversas configurações, chegamos a uma taxa aceitável de transição. Assim, utilizamos uma taxa de corte na frequência de 2,4 *frames*, onde a cada 10 *frames* selecionamos um, considerando a percepção humana com mínimo de 23,97 FPS.

A construção do *dataset* ocorreu em duas etapas de *deduplication* dos dados. A primeira etapa realizada antes de efetuar o corte dos vídeos, analisando a duração do vídeo e tamanho do arquivo, os arquivos duplicados foram removidos. Após a extração dos vídeos em *frames*, foi efetuado uma segunda etapa de *deduplication*, no qual foi gerado um *hash* MD5 de cada um dos *frames* extraídos, permitindo efetuar a remoção de *frames* idênticos. Por fim, cada amostra de *frame* foi rotulada manualmente e individualmente como *Pornô*, *Normal* ou *Contexto Pornô*. As amostras dos *frames de Contexto Pornô* são *frames* extraídos de vídeos pornográficos, mas sem conteúdo pornográfico. Por outro lado, as amostras dos *frames Normais* são extraídas de vídeos não pornogáficos, enquanto as amostras dos *frames Pornô* são obtidas dos vídeos pornográficos com conteúdo explicitamente pornográfico. Por fim, para facilitar a avaliação foi efetuado um balanceamento entre as classes, a estrutura da composição do *dataset* pode ser observado na Tabela 3.

Tabela 3 - Estrutura do *dataset*

Estrutura *Frames Dataset*					
Vídeos	***Frames***			**Amostras**	
	Gerados	***Rótulos***	***Selecionados***	***Pornô***	***Normal***
Vídeos Pornográficos	237426	***Pornô***	211201	211201	-
		Contexto Pornô	26225	-	26225
Vídeos Normais	239056	***Normal***	184976	-	184976
Total Frames	476482	-	422402	211201	211201

Para fins de avaliação, o *dataset* proposto foi dividido em três subconjuntos: treinamento, validação e teste, compreendendo 60%, 20% e 20%, respectivamente, dos vídeos. Portanto, cada subconjunto contém vídeos exclusivos, permitindo a avaliação adequada da generalização da CNN de acordo com o contexto do vídeo. Evitando assim que os *frames* de um mesmo vídeo estejam contidos simultaneamente em mais de um subconjunto (treinamento, teste e validação).

5.2. Treinamento CNN

Os experimentos com as CNN aqui transcritos foram realizados utilizando o framework para *Deep Learning* denominado Caffe. A rede CNN foi treinada com diferentes arquiteturas, Alexnet, Caffenet e Googlenet. Primeiramente utilizando a abordagem tradicional, gerando um modelo para cada uma das arquiteturas.

Posteriormente as arquiteturas foram treinadas utilizando *Transfer Learning*, onde foram utilizados modelos já treinados no *dataset* ImageNet (Jia, et al., 2014), reaproveitando os pesos como ponto de partida para gerar os novos modelos. Para validação da proposta foram gerados ao todo seis modelos CNN. O conjunto de teste foi utilizado sobre os modelos anteriormente treinados.

5.3. Detecção de Pornografia em Vídeo

Avaliamos o novo *dataset* com os seis modelos anteriormente treinados. Onde cada arquitetura da CNN (Googlenet, Alexnet e Caffenet) foi avaliada com e sem aplicação do *Transfer Learning*.

As CNNs foram treinadas usando o conjunto de dados de treinamento e avaliadas ao longo de sua fase de treinamento usando o conjunto de dados de validação. A precisão final da CNN é medida usando o conjunto de dados de teste. Para a tarefa de treinamento, as CNNs foram treinadas considerando duas classes: *Normal* e *Pornô*. Nesse caso, a classe *Normal* é composta pelos *frames Normal* e *Contexto Pornô*. Esse procedimento foi adotado porque a CNN, quando usada em aplicativos de tempo real, deve rotular os *frames* de *Contexto Pornô* como *Normal*, pois eles não apresentam nenhum tipo teor de pornográfico. Considerando que a quantidade de *frames* de *Contexto Pornô* é significativamente menor a quantidade *de frames Normal* e *Pornô,* observa-se um desbalanceamento nessa etapa inicial. A fim de evidenciar melhor o problema de contexto, foi aplicada uma técnica de *under-sampling* que efetua um balanceamento do *dataset* focado na classe majoritária, eliminando aleatoriamente instâncias da classe com maior número de ocorrências (*frames* de *Contexto Pornô* como *Normal*).

Conforme protocolo estabelecido para cada CNN, antes da fase de treinamento foi realizada uma sequência de experimentos, permintindo ajustar de forma empírica o valor adequado do parâmetro *learning rate* destinado a cada arquitetura. No processo de ajuste para cada valor testado foram executadas 500 iterações, por default do *framework* Caffe, foi utilizado o algoritmo de otimização SGD (*Stochastic Gradient Descent*), efetuando a atualização dos pesos por uma combinação linear do gradiente negativo e a atualização do peso anterior. Posteriormente, as arquiteturas foram efetivamente treinadas, onde foi utilizado o valor do *learning rate* que obteve o menor valor para o parâmetro *loss,* o que correspondente a cada uma das arquiteturas. O parâmetro *momentum* foi fixado em 0.9 , permitindo obter uma melhor convergência (Sutskever, et al., 2013).

Na Tabela 4, podemos verificar a acurácia de cada uma das arquiteturas testadas. Os resultados estão dispostos em três classes (*Normal*, *Pornô* e *Contexto Pornô*) possibilitando ressaltar bem o problema de contexto. Observe que a acurácia obtida para as classes dos *frames Pornô* e *Contexto Pornô* são bastante inferiores quando comparadas com os *frames* da classe *Normal*. Tal fato, ocorre devido a ambos os *frames* corresponderem aos *frames* advindos do mesmo vídeo. Denota que a CNN teve certa dificuldade em determinar exatamente a qual classe o *frame* pertence, refletindo em uma acurácia baixa para os *frames Pornô* e *Contexto Pornô*.

Tabela 4 - Acurácia por classe de *frame*

Arquiteturas	**Acurácia por *frame* (%)**		
	Normal	***Pornô***	***Contexto Pornô***
Caffenet	94,58	68,52	65,86
Alexnet	97,80	63,14	72,85
Googlenet	95,48	62,07	71,57
Caffenet com *Transfer Learning*	99,11	79,52	69,73
Alexnet com *Transfer Learning*	96,92	73,02	65,76
Googlenet com *Transfer Learning*	95,46	62,07	71,57

Conforme verificado, existe um problema de contexto em relação à classificação dos *frames* dos vídeos pornográficos. Na nossa opinião, a CNN não foi capaz de abstrair todas as características necessárias para determinar o contexto do vídeo. Ocorrência comprovada pelo padrão da acurácia obtida em todos os modelos (Alexnet, Caffenet e Googlenet), tanto de maneira tradicional como por meio da utilização de *Transfer Learning*, a tendência da acurácia entre os modelos foi similar. Assim podemos aferir que a CNN possui limitações para determinar o contexto do vídeo analisando um único *frame* por vez.

No gráfico da Figura 14 são apresentadas as taxas de TP permitindo avaliar a precisão dos modelos na classificação no *dataset* proposto para cada uma das classes presentes no conjunto de dados. É possível observar que as arquiteturas CNN foram capazes de detectar adequadamente os *frames* normais, atingindo até 99,1% da taxa de verdadeiro positivo para o Caffenet com *Transfer Learning*, por exemplo.

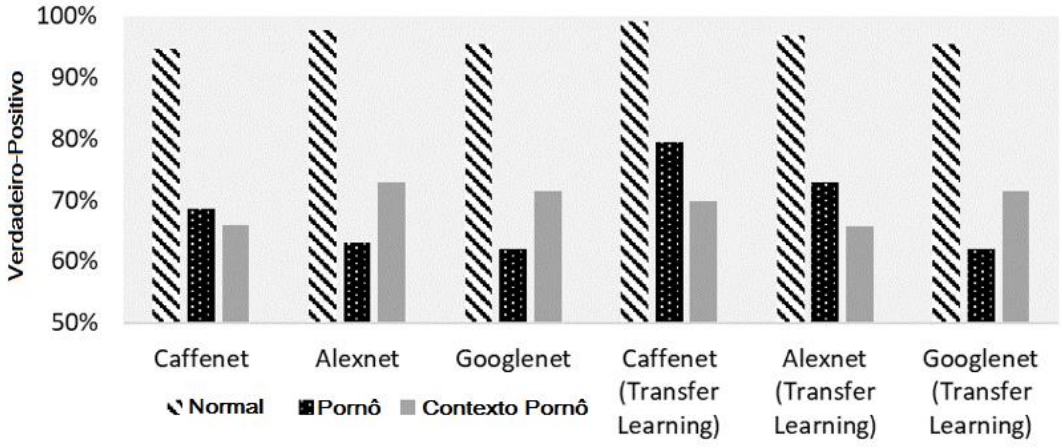

Figura 14 - Avaliação da classificação das arquiteturas tradicionais no *dataset* proposto

Por outro lado, na detecção de *frames Pornô* e de *Contexto Pornô* é significativamente menor quando comparada aos *frames Normais*. Em relação aos *frames Pornô*, a taxa de detecção variou de 62% a 79%, para o Googlenet e Caffenet com *Transfer Learning*, respectivamente. Da mesma forma, os *frames* de *Contexto Pornô* também apresentaram resultados semelhantes, variando de 65% a 72%, para o Caffenet e Alexnet, respectivamente.

A Tabela 5 dispõem dos resultados da matriz de confusão das arquiteturas CNN testadas sobre o *dataset* balanceado com *under-sampling*, assim é possível entender o conflito entre as classes. Nesta tabela são apresentadas as três classes, contendo o erro e acerto correspondentes a cada uma delas. Em critérios de avaliação, a classe do *frame Contexto Pornô* é avaliada como não pornográfico.

Tabela 5 - Matriz de confusão do *dataset* proposto com *under-sampling*

Matriz de Confusão						
Arquiteturas	***Normal***		***Pornô***		***Contexto Pornô***	
	Sim	**não**	**sim**	**não**	**sim**	**não**
Caffenet	4978	285	3606	1657	3466	1797
Alexnet	5147	116	3323	1940	3834	1429
Googlenet	5025	238	3267	1996	3767	1496
Caffenet com *Transfer Learning*	5216	47	4185	1078	3670	1593
Alexnet com *Transfer Learning*	5101	162	3843	1420	3461	1802
Googlenet com *Transfer Learning*	5025	239	3267	1996	3767	1496

A acurácia diminuiu para ambos exemplos de amostras de *frames Pornô* e de *Contexto Pornô*, ocorridas onde ambos exemplos de amostras de *frames* foram gerados a partir do mesmo vídeo. No entanto, apesar de serem do mesmo vídeo, os *frames*

amostrados nem sempre exibiam o conteúdo com classificação pornográfica. Portanto, devido à falta de contexto durante a classificação dos *frames* amostrados, os *frames* individuais podem ser classificados incorretamente.

Além disso, quando utilizado em aplicativos de tempo real, o mecanismo de classificação deve ser capaz de lidar com o conteúdo pornográfico de acordo com seu contexto. Consequentemente, em vez de rotular o vídeo inteiro como pornográfico, o mecanismo de detecção deve poder classificar corretamente apenas uma parte dele.

5.4. Avaliação e Análise dos Métodos

A avaliação visa responder às seguintes questões de pesquisa (RQ): i) *como a abordagem Stacking Meta Learner melhora a acurácia da detecção?* ii) *qual é o tamanho ideal de cena para a detecção de conteúdo pornográfico baseada no contexto?* iii) *qual é o melhor classificador superficial para detecção de conteúdo pornográfico baseada no contexto?* iv) *qual é o desempenho do Stacking Meta Learner projetado em conjuntos de dados do estado da arte?* v) *A métrica de leveza auxilia na criação de uma CNN otimizada para a implemetação em dispositivos com recursos limitados?* vi) *A técnica de amostragem de frames afeta a precisão do modelo?* e vii) *A técnica de verificação ajuda na melhoria da acurácia do modelo?*

Entre os experimentos realizados para avaliação, foi efetuada primeiramente uma comparação dos modelos utilizando apenas CNN *versus* a abordagem proposta. Em relação às arquiteturas CNN, foram utilizadas a Alexnet, Caffenet e Googlenet. Testando assim os modelos treinados tanto tradicionalmente como por meio de uma técnica de *Transfer Learning*. Os modelos previamente treinados foram utilizados na estruturação da abordagem *Stacking Meta Learner* (organizados em sequência em uma janela deslizante de *frames*). Para os experimentos, o tamanho da janela deslizante foi configurado com 21 *frames*, posteriormente aplicado o classificador raso Naive Bayes sobre cada trecho da cena percorrido pela janela deslizante. A avaliação da abordagem realizada sobre os módulos está em destaque na Figura 15.

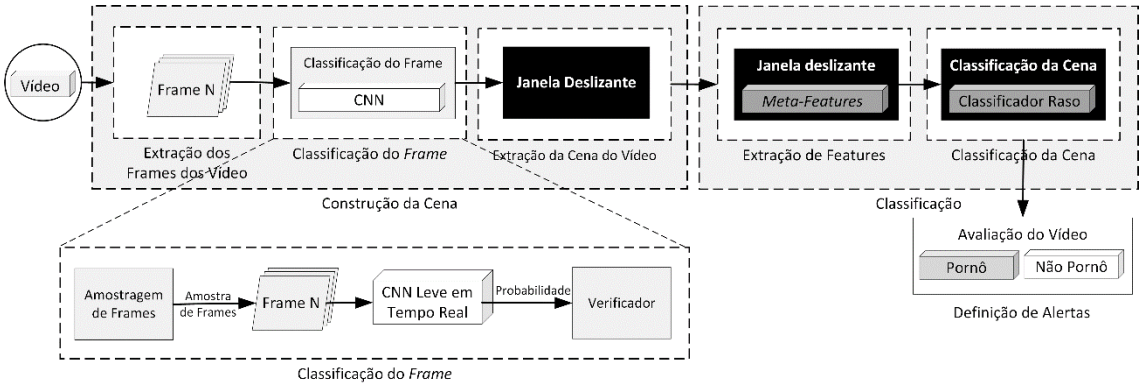

Figura 15 - Avaliação módulo *Stacking Meta Learner*

Os resultados da acurácia de cada um dos modelos podem ser visualizados na Tabela 6. Analisando os resultados, observa-se o aumento da acurácia para todos os modelos, chegando a alcançar um incremento superior a 10% para modelo treinado com Googlenet utilizando *Transfer Learning*.

Tabela 6 - Acurácia modelo CNN versus *Stacking Meta Learner*

Arquiteturas	**Acurácia dos modelos (%)**	
	Tradicional	***Proposta***
Caffenet	76,32	81,35
Alexnet	77,93	83,68
Googlenet	76,38	83,46
Caffenet com *Transfer Learning*	82,78	88,18
Alexnet com *Transfer Learning*	78,57	84,05
Googlenet com *Transfer Learning*	76,38	87,74

Deve-se considerar o impacto da arquitetura CNN na estruturação da abordagem *Stacking Meta Learner*, onde a primeira etapa da abordagem utiliza das arquiteturas CNN para a extração do vetor de características, influenciando indiretamente na acurácia obtida, pois, ao selecionar um modelo CNN que seja impreciso, as características absorvidas talvez não sejam totalmente suficientes para aplicação na abordagem *Stacking Meta Learner*. Entretanto, isso não pode ser aplicado em vias gerais de regras, sendo que a nossa abordagem visa analisar o contexto de maneira mais ampla. Efetuando a análise de um conjunto maior de *frames*, a acurácia obtida por nossa proposta tende a ser mais precisa. Isso pode ser observado em relação ao ganho de acurácia obtido entre um modelo e outro. Verificando os resultados na tabela, observamos que tal ganho não é aplicado maneira gradual.

Conforme já mencionado, o *Stacking Meta Learner* alcançou um aumento significativo em relação ao ganho de acurácia. Contudo, ainda sim identificamos a possibilidade de aprimorar o valor da acurácia e, para tanto, vislumbramos a adoção dos métodos propostos nas secções subsequentes.

5.4.1. Avaliação da Abordagem Stacking Meta Learner

A presente seção tem como objetivo responder a seguinte RQ1: *"Como a abordagem Stacking Meta Learner melhora a acurácia da detecção?"*. Para responder a essa questão de pesquisa foi utilizado o método proposto com Naive Bayes como classificador raso. Na Figura 16 é mostrada uma comparação da taxa de TP e TN entre a abordagem proposta e a CNN tradicional.

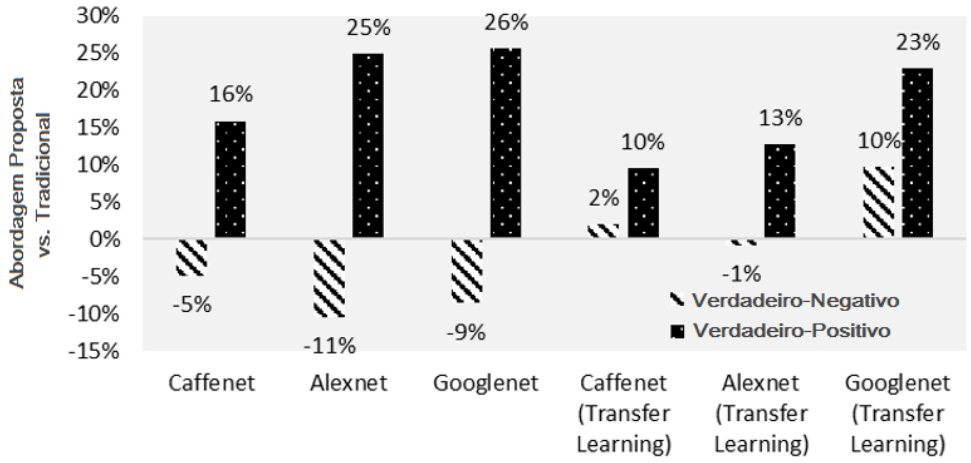

Figura 16 - Acurácia abordagem *Stacking Meta Learner* versus CNNs tradicionais sob o *dataset* proposto

É possível observar que para abordagem *Stacking Meta Learner* utilizando CNNs com *Transfer Learning* a técnica proposta aumentou significativamente a taxa de TP na detecção de amostras de *frames Pornô* (Seção 5.2), além de melhorar a taxa de TN na detecção das amostras dos *frames Normal* e *Contexto Pornô* para o Caffenet e Googlenet. Em relação às CNNs sem *Transfer Learning*, a abordagem proposta também aumentou significativamente a taxa de TP. Um aumento de até 26%, em contraste, diminuiu a taxa de FN em até 11%, no pior caso. Consequentemente, a abordagem proposta melhorou significativamente a taxa de detecção, avaliando a cena do *frame* pivô. Em outras palavras, a avaliação da cena em que um *frame* com pornografia ocorre fornece taxas mais altas de acurácia.

5.4.2. Avaliação do Tamanho Janela de Frames

A presente seção tem como objetivo responder a seguinte RQ2: *"Qual é o tamanho ideal de cena para a detecção de conteúdo pornográfico baseada no contexto?"*. Para encontrar o tamanho ideal da cena (Figura 9, Janela deslizante de *frames*) para detecção, vários tamanhos de cena, de 1 a 100 *frames*, foram avaliados em relação à acurácia obtida e à troca de atraso na detecção. A avaliação infere os resultados obtidos sobre o módulo da extração da cena em destaque na Figura 17.

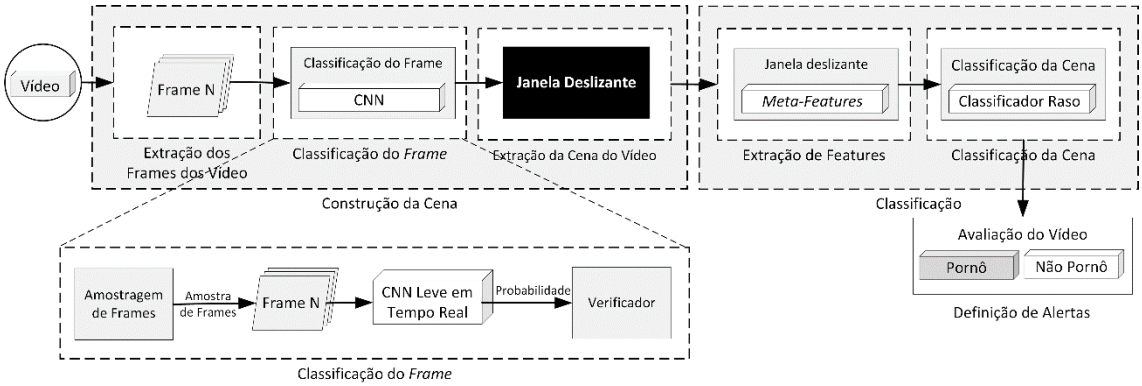

Figura 17 - Módulo extração de cena / janela deslizante de *frames*

As janelas deslizantes com tamanhos de cenas mais altos, incorrem em maior atraso de detecção causado pela análise dos *frames* anteriores e posteriores do *frame*-pivô. A Figura 18 dispõe do gráfico dos experimentos realizados com diferentes configurações do tamanho da janela de *frames* e seu impacto sob a acurácia.

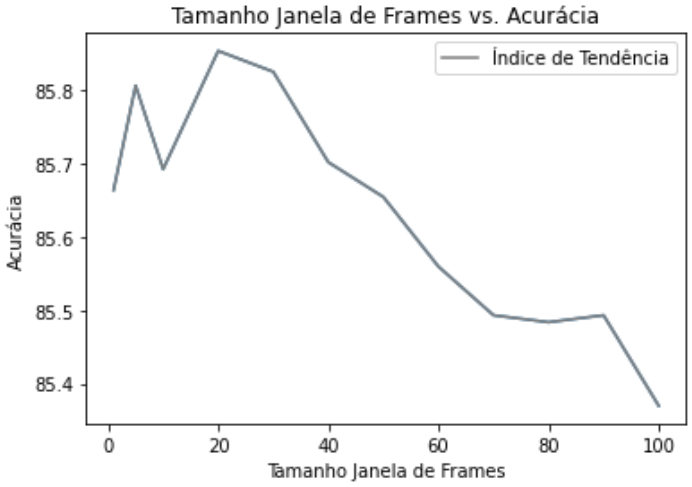

Figura 18 - Seleção tamanho janela de *frames*

Os resultados obtidos indicaram que uma cena superior a 21 *frames* não melhora significativamente a acurácia. Por outro lado, um tamanho de cena menor afeta

significativamente a acurácia. Portanto, o tamanho ideal da cena (buffer de amostra de *frames*) foi estabelecido como 21 *frames*, compreendendo: 1 pivô, 10 *frames* anteriores e 10 *frames* posteriores. Em outras palavras, devido à taxa de amostragem de *frames* (intervalo de ~ 0,3 segundo), a abordagem proposta exige um tamanho de cena de ~ 3 segundos (ou seja, 21 amostras de *frames* armazenados em buffer na janela deslizante). Consequentemente, apresenta um atraso de detecção inicial de 1,5s (isto é, tempo para armazenar em buffer as 10 amostras de *frames* posteriores, dado que o *frame*-pivô representa os primeiros 20 *frames* de vídeo a serem mostrados ao usuário).

5.4.3. Avaliação do Melhor Indutor Raso

O módulo de classificação *Stacking Meta Learner* recebe os arquivos "*.arff"* do módulo de extração. Nesse módulo os arquivos ".arff" foram executados no WEKA, um ambiente de aprendizagem de máquina e mineração de dados. Para responder a RQ3: "*Qual é o melhor classificador superficial para detecção de conteúdo pornográfico baseada no contexto?"*, diversos testes em classificadores rasos foram realizados, e a avaliação foi realizada sobre o módulo classificação da cena, em específico sobre o classificador raso em destaque na Figura 19.

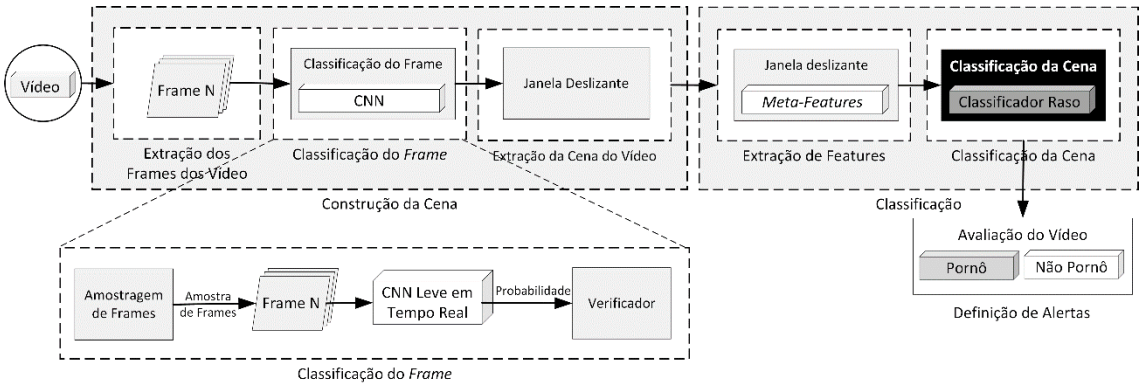

Figura 19 - Módulo classificação da cena / classificador raso

Entre os principais algoritmos de aprendizagem de máquinas utilizados, destacam-se o Adaboost, IBK, J48, NaiveBayes e o Random Forest. O conjunto completo dos algoritmos testados no Weka segue disposto na seção de anexos. Conforme apresentado na Tabela 7, podemos observar que o classificador que obteve melhor acurácia foi o NaiveBayes, sendo que o único ajuste de parâmetros realizado nesse indutor foi o *useKernelEstimator* setado com o valor *true*. Nos demais indutores a alteração dos parâmetros não foi significativa.

Tabela 7 - Acurácia *Stacking Meta Learner* por classificador

Acurácia dos Classificadores (%)					
Modelos	**Modelos Tradicionais**				
	Adaboost	***KNN***	***C4.5***	***NaiveBayes***	***RandomForest***
Caffenet	76,32	76,35	76,32	81,34	76,33
Alexnet	77,93	78,44	78,19	83,68	78,58
Googlenet	77,48	81,45	79,99	83,46	79,42
Modelos	**Modelos com *Transfer Learning***				
	Adaboost	***KNN***	***C4.5***	***NaiveBayes***	***RandomForest***
Caffenet	78,57	87,03	78.57	88,18	79,12
Alexnet	78,57	82,54	78.57	84,04	79,12
Googlenet	83,09	87,02	83.10	87,74	83,85

O Naive Bayes alcançou 88,18% da acurácia com o Caffenet utilizando *Transfer Learning*. Consequentemente, o classificador raso Naive Bayes foi utilizado nas avaliações posteriores.

5.4.4. Validação do Método no Dataset Pornography-2k

A fim de validar o método proposto, bem como posicionar o mesmo em relação aos trabalhos dispostos no estado da arte, os experimentos foram replicados sobre o *dataset Pornography-2k*. O *dataset Pornography-2k* (Perez, et al., 2017) é composto por um conjunto de dois mil vídeos, rotulados metade como pornográficos e a outra metade como não pornográficos. A presente seção tem como objetivo responder à seguinte RQ4: *"Qual é o desempenho do Stacking Meta Learner projetado em conjuntos de dados do estado da arte?"*.

É imprescindível salientar a diferença entre os *datasets*. O *dataset* disponibilizado anteriormente é um *dataset* de granularidade fina, e foi rotulado *frame* a *frame,* justamente visando à identificação do contexto dos *frames*. Já o *dataset Pornography-2k* possui granularidade graúda, pois nessa abordagem a rotulagem é realizada sobre vídeos (pornográficos ou não pornográficos), ou seja, os *frames* dos vídeos herdam o rótulo que lhe foi atribuído ao vídeo. Porém é necessário ressaltar que nos vídeos pornográficos existem *frames* que talvez não sejam de caráter pornográfico, o que acaba caracterizando o problema de contexto. Na Figura 20 é bem exposto o problema de contexto, em que se observa a retirada de ambas amostras do mesmo vídeo – a primeira corresponde a uma amostra de frame classificada como *Contexto Porno*, também todos

os *frames* que a antecedem não possuem teor pornográfico. Por sua vez, a segunda amostra foi classificada como *frame Pornô* corresponde ao seu conteúdo impróprio.

Figura 20 – Problema de contexto no *datasets*

Utilizando uma abordagem de granularidade graúda ambos os *frames* serão rotulados como *frames* pornográficos. Para melhor compreensão é exposta na Figura 21 uma amostra de rotulagem de um vídeo pornográfico, demonstrando as diferenças de granularidade na rotulagem realizadas sobre vídeo e *frame*.

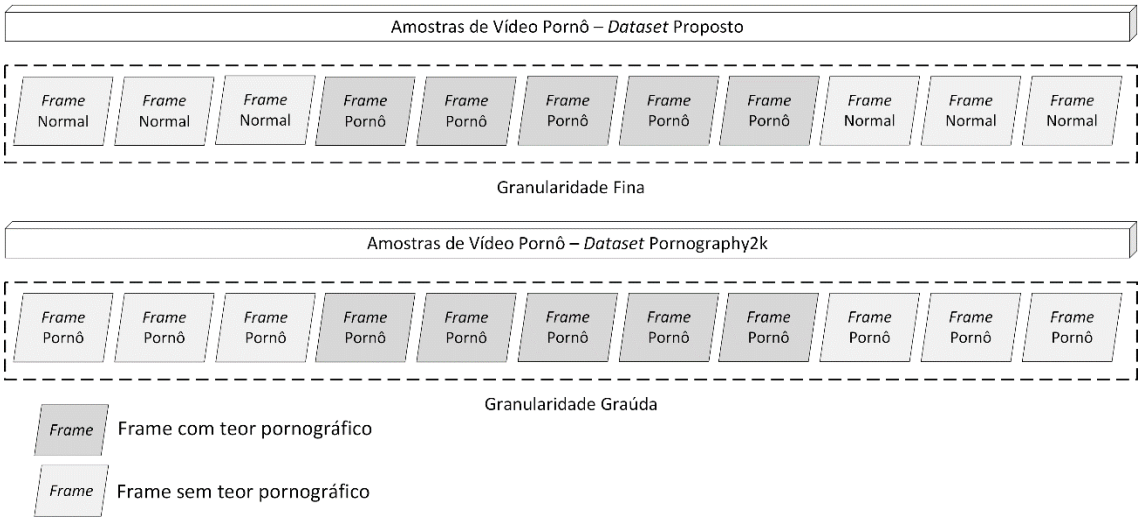

Figura 21 - Granularidade *datasets*

A rotulagem em granularidade fina possui vantagens em relação à granularidade graúda, principalmente quando relacionadas com aplicações de análise *frame* a *frame*. Tal como aplicações de vídeo em tempo real, onde é necessário a análise *frame* a *frame* antes de realizar a transmissão.

Os experimentos da abordagem *Stacking Meta Learner* foram replicados sobre o *dataset Pornography-2k*. Primeiramente, a CNN foi treinada sobre o *dataset Pornography-2k* de maneira convencional, sendo utilizadas as arquiteturas Alexnet,

Caffenet e Googlenet. Posteriormente, treinamos a rede sobre as mesmas arquiteturas utilizando *Transfer Learning*. Na sequência, foi aplicada a abordagem *Stacking Meta Learner* sobre os modelos gerados.

Os resultados obtidos sobre o *dataset Pornography-2k* são expostos na Tabela 8, onde foi efetuada uma avaliação do *dataset* utilizando uma visão de granularidade fina, ou seja, a análise da abordagem efetuada sobre a granularidade de *frames*.

Tabela 8 - Acurácia modelo CNN versus *Stacking Meta Learner* (*dataset Pornography-2k*)

Arquiteturas	**Tradicional**		**Proposta**	
	Acurácia (%)	***F-Measure***	**Acurácia (%)**	***F-Measure***
Caffenet	87,97	0,87	89,11	0,91
Alexnet	87,56	0,88	89,20	0,90
Googlenet	91,51	0,89	92,45	0,94
Caffenet com *Transfer Learning*	92,89	0,90	94,00	0,95
Alexnet com *Transfer Learning*	90,30	0,86	91,47	0,94
Googlenet com *Transfer Learning*	95,19	0,95	96,32	0,98

Conforme os resultados obtidos, tanto no *dataset* proposto como no *dataset Pornography-2k,* a abordagem *Stacking Meta Learner* comparada com uma abordagem CNN tradicional demostrou ser superior. Observa-se uma tendência em todos os modelos treinados. Indícios comprovaram uma melhoria em todos os modelos, tanto em termos de acurácia quanto para o valor de *F-Measure*.

É necessário frisar que o *dataset Pornography-2k* não foi rotulado *frame* a *frame* e mesmo assim a abordagem *Stacking Meta Learner* tem conseguido alcançar alguns resultados interessantes. Comparando os resultados de ambos os *datasets* foi possível verificar que abordagem utilizando janela de *frames* é mais eficaz em um *dataset* de granularidade fina, fator identificado devido às características intrínsecas no *dataset*, onde a rotulagem foi realizada *frame* a *frame*.

Para efetuar o cálculo das métricas, bem como para comparar os experimentos com os trabalhos dispostos no estado da arte, o protocolo pré-estabelecidos pelos autores (Perez, et al., 2017) foi empregado, assim como para o cálculo do *F-Measure* foi adotado a Equação 10:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall} \quad (10)$$

Para atribuir peso a classe pornográfica, os autores definem o valor de β sendo igual a dois.

Considerando que o conjunto de dados *Pornography-2k* foi rotulado em uma granularidade de vídeo, consequentemente para fins de *benchmarking*, para esse *dataset* a abordagem *Stacking Meta Learner* foi avaliada em relação à precisão da classificação em uma granularidade de vídeo. Para atingir esse objetivo, um vídeo é considerado vídeo *Pornô* se a maioria de seus *frames* forem rotulados como *Pornô*.

Além disso, utilizar a granularidade de vídeo sobre o *dataset Pornography-2k* permite que a técnica *Stacking Meta Learner* seja comparada com a abordagem proposta por Moreira *et al*. (Moreira, et al., 2019).

A Tabela 9 dispõe dos resultados da acurácia obtida no conjunto de dados *Pornografia-2k*. É possível notar que a técnica proposta alcançou resultados semelhantes quando comparada às abordagens de detecção do estado da arte. Além disso, a técnica proposta foi capaz de melhorar a medida de *F-Measure* ao aplicar o Googlenet com *Transfer learning*, ocasionando apenas 0,25% de *tradeoff*[14] na acurácia.

Tabela 9 - Comparação da abordagem *Stacking Meta Learner* no *dataset* do estado da arte *Pornography2k*

Técnica de Classificação		**Acurácia**	***F-Measure***
Abordagem Proposta	***Caffenet (Transfer Learning)***	94,50%	95,73%
	Alexnet (Transfer Learning)	93,73%	94,84%
	Googlenet (Transfer Learning)	95,75%	96,82%
Moreira et al [2018] (*Técnica estática*)		96,00%	96,10%

5.5. Análise da Otimização de Recursos

A análise de otimização de recursos foi estruturada visando responder às três últimas questões de pesquisa (RQ), apresentando os critérios de avaliação e os resultados obtidos.

Para fins de avaliação, consideramos um cenário em que um usuário de dispositivo com recursos limitados pretende detectar conteúdo pornográfico em tempo real em vídeos. Avaliação realizada sobre os módulos em destaque na Figura 22. Para esse fim, foi utilizado o *dataset Pornography2k* (Perez, et al., 2017). A periodicidade de

[14] Tradeoff - define uma situação em que há conflito de escolha, caracteriza-se em uma ação econômica que visa à resolução de problema, entretanto acarretando outro, obrigando a uma escolha.

amostragem de *frames* padrão é definida em 0,3 FPS, semelhante aos trabalhos relacionados (Ng, et al., 2015). Para fins de avaliação, seguimos o mesmo protocolo definido para o *Stacking Meta Learner*, sendo o conjunto de dados foi dividido em três partes: treinamento, validação e teste – onde cada parte compreende 60%, 20% e 20% dos vídeos pornográficos e normais, respectivamente. Consequentemente, os *frames* de cada conjunto de dados compreendem vídeos distintos, tanto para classe de vídeo normal quanto pornográfico.

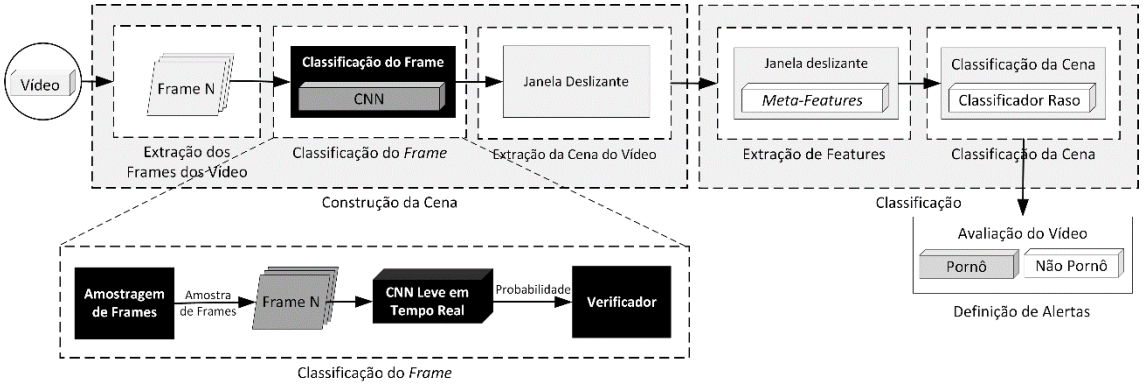

Figura 22 - Módulos de otimização

As arquiteturas CNNs modificadas do Caffenet (disposto na Secção 4.2.2) também foram implementadas no framework Caffe. Cada arquitetura foi treinada utilizando o conjunto de dados de treinamento e validação. Por fim, os resultados obtidos correspondem ao conjunto de testes. Semelhante ao realizado anteriormente, cada arquitetura CNN foi treinada sob um ciclo de 50.000 iterações. Foi utilizando o *learning rate* previamente selecionado para cada uma das arquiteturas treinadas que testes foram realizados empiricamente empregando ciclos com 500 iterações, permitindo definir o melhor valor do parâmetro *learning rate* para o efetivo treinamento da CNN (Wei, et al., 2016). Semelhante aos trabalhos relacionados, todas as camadas de entrada do Caffenet são compostas por um tamanho de matriz de 256x256 (Nogueira, et al., 2017). O equipamento utilizado para a execução desse teste foi um Ubuntu Desktop 16.04 equipado com um Intel i7, 16GB de memória e uma GPU Nvidia Titan XP. As configurações foram realizadas sobre o conjunto de dados de validação e posteriormente avaliadas utilizando o conjunto de dados de teste.

5.5.1. Avaliação da Métrica de Leveza

A presente seção tem como objetivo responder à seguinte RQ5: “A métrica de ‘leveza’ auxilia na criação de uma CNN otimizada para a implementação em dispositivos com recursos limitados?”. Essa RQ tem como propósito avaliar se o uso da medida leveza proposta (Seção 4.2.2) na fase de construção da arquitetura CNN pode melhorar as arquiteturas atuais da CNN. Avaliação realizada na abordagem sobre o módulo *CNN leve em tempo real* em destaque na Figura 12 (Figura 12 - Módulo *CNN leve em tempo real*).

Para esse fim, o layout da arquitetura CNN Caffenet (Jia, et al., 2014) foi adaptado em relação à métrica de leveza. O objetivo é criar uma arquitetura CNN personalizada, construída levando em consideração a métrica de leveza. Para atingir esse objetivo, a arquitetura Caffenet foi dividida em 8 partes, cada uma composta por camadas específicas, como mostra a Figura 23.

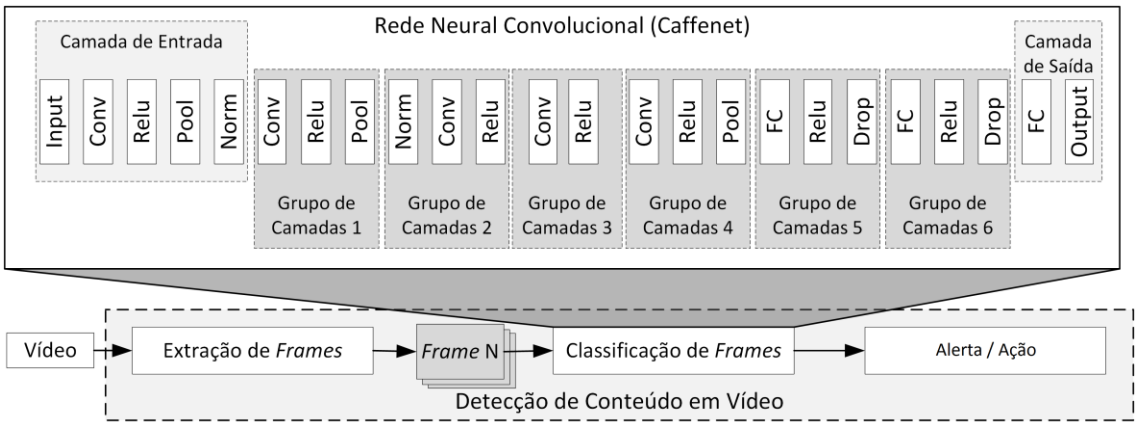

Figura 23 - Processo típico de detecção de conteúdo de vídeo

As camadas da CNN foram agrupadas respeitando seus propósitos e, portanto, cada grupo executa uma sequência de operações normalmente encontrada em uma arquitetura CNN. Consequentemente, o tamanho da CNN pode ser diminuído com a remoção de grupos específicos sem danificar a arquitetura CNN. O objetivo é realizar modificações típicas feitas em arquiteturas CNN mais leves (compactas e exigindo menor custo computacional). Além disso, as camadas de entrada e saída são sempre usadas na arquitetura CNN, enquanto a outra camada dentro dos grupos 1 a 6 pode ser usada ou não. Devido à assimetria dos dados, para o cálculo da métrica de leveza, cada uma das medidas: acurácia, taxa de transferência e memória foram normalizadas usando o processo de normalização *Z-Score*, disposto conforme o cálculo da Equação 9.

A Tabela 10 mostra a acurácia obtida, os requisitos de memória, a taxa de transferência de detecção e a métrica leveza quando as camadas de grupo são adicionadas ou removidas. É possível observar uma troca significativa de memória, acurácia e taxa de transferência, quando as camadas são removidas ou adicionadas. A melhor métrica de leveza da CNN foi obtida usando os grupos de camadas 1, 2, 3 e 4. Foi capaz de atingir 89,30% de acurácia, com apenas 0,67% de troca de acurácia pela CNN mais precisa, exigindo apenas 1,21% da memória e atingindo 270% a mais de taxa de transferência de detecção. Ao comparar com a CNN mais rápida, apresentou 123 menos na taxa de transferência de *frames*/seg, alcançando uma acurácia de 2,22% a mais e exigindo apenas 7,3 MB a mais de memória.

Tabela 10 - Métrica de "*leveza*" das técnicas de detecção de conteúdo de vídeo baseadas em CNN.

Grupo de Camadas Arquitetura CNN Caffenet						**Acurácia (%)**	***FPS***	**Memória *(MB)***	**Métrica de Leveza**
1	2	3	4	5	6				
✓	✓	✓	✓	✓	✓	88,03	63,97	217	-0,41
	✓	✓	✓	✓	✓	89,97	81,37	746	-0,38
		✓	✓	✓	✓	88,92	91,54	743	-0,88
			✓	✓	✓	89,16	98,57	741	-0,68
				✓	✓	84,51	68,25	1200	-4,54
					✓	84,48	73,01	1100	-4,30
✓	✓	✓	✓	✓		89,47	19,21	153	1,59
✓	✓	✓	✓			89,30	301,69	9	2,64
✓	✓	✓				88,33	304,70	7,8	2,12
✓	✓					88,37	343,62	5,2	2,46
✓						87,08	423,30	1,7	2,38

Consequentemente, é possível observar que a métrica leveza ajuda na implantação da CNN para dispositivos com recursos limitados. A medida proposta permite encontrar a arquitetura CNN ideal que melhora o conjunto de propriedades desejadas, precisão, taxa de transferência de detecção e uso de memória, possibilitando a implementação adequada da arquitetura CNN para dispositivos com recursos restritos. O conjunto completo dos testes realizados para obtenção da métrica de leveza estão disponibilizados na secção de anexos.

A fim avaliar o impacto da modificação das arquiteturas CNN em dispositivos com recursos limitados, foi efetuada uma série de experimentos de *benchmark*. A Tabela 11 dispõe dos resultados dos modelos previamente treinados sobre o *dataset* Pornography2k avaliados utilizando o conjunto de testes em um dispositivo de recurso limitado. Para tal utilizamos a placa de desenvolvimento da NVIDIA Jetson TK1. Assim sendo, foi possível replicar o comportamento computacional similar a um dispositivo

móvel, observando que a mesma especificação do processador e placa de vídeo disposta no kit é utilizada em alguns smartphones e tablets.

Tabela 11 - Impacto das arquiteturas CNN aplicadas em um dispositivo com recurso limitado

Camadas	***Tamanho (MB)***	**Acurácia (%)**	***TK1 GPU (GK20A)***	***CPU TK1***
Todas	217	88,03	534m57.975s	3763m15.177s
1,3,4,5,6,7 e 8	746	Exceção*	Exceção*	Exceção*
1,4,5,6,7 e 8	743	Exceção*	Exceção*	Exceção*
1,5,6,7 e 8	741	Exceção*	Exceção*	Exceção*
1,6,7 e 8	1200	Exceção*	Exceção*	Exceção*
1,7 e 8	1100	Exceção*	Exceção*	Exceção*
1,2,3,4,5,6 e 8	153	89,47	563m49.390s	3275m32.686s
1,2,3,4,5 e 8	9,0	89,30	420m42.590s	2410m17.557s
1,2,3,4 e 8	7,8	88,33	451m37.664s	2181m39.851s
1,2,3 e 8	5,2	88,37	418m28.984s	2044m25.446s
1,2 e 8	1,7	87,08	375m7.981s	1384m30.661s
Legenda: Exceção* - Limite de memória excedido				

Conforme observado, nem todas as arquiteturas CNN foram possíveis de serem executadas. O Jectson TK1 não possui recurso suficiente para carregar os modelos maiores, e tais modelos excedem o limite de recurso de memória disponível. Os experimentos foram realizados tanto utilizando apenas o processador TK1 como também utilizando a GPU GK20A. Selecionando a melhor arquitetura com a métrica de leveza (disposto na Tabela 9), conseguimos classificar na média 10,23 *frames* por segundo. Os experimentos estruturados sobre o *dataset Pornography2k*, bem como os experimentos de *benchmark* estão dispostos na íntegra na seção de anexos.

5.5.2. Avaliação da Técnica de Amostragem de Frames

A presente seção tem como objetivo responder à seguinte RQ6: “A técnica de amostragem de *frames* afeta a acurácia do modelo?”. Essa RQ tem por propósito avaliar a troca (tradeoff) de acurácia do modelo causada pela diminuição do número de *frames* usados para fins de treinamento e avaliação, tendo como resultado a melhora do *throughput* da detecção (consulte a Seção 4.4.1). Avaliação realizada na abordagem sobre o módulo de *amostragem de frames* em destaque na Figura 11 (Figura 11 - Módulo *amostragem de frames*). Para esse fim, as amostras de *frames* são escolhidas reativamente para treinamento e avaliação. Em outras palavras, os *frames* de vídeo são selecionados em intervalos de tempo específicos, em vez de escolher *frames* específicos, como a

utilização de *frames* chaves. O motivo é que a amostragem de *frames* visa aumentar o *throughput* da detecção. Portanto, a escolha reativa dos *frames* permite diminuir ainda mais o processamento exigido. Consequentemente, as arquiteturas CNN são treinadas e avaliadas em relação à periodicidade da amostragem de *frames* escolhida.

Na Figura 24 é mostrada a relação entre a acurácia da arquitetura CNN CaffeNet, mais especificamente as taxas TN (True Negative) e TP (True Positive), e o intervalo de amostragem de *frames*. É possível notar que o aumento do intervalo de amostragem de *frames* não apresenta efeito significativo na acurácia do modelo. Por exemplo, ao usar um atraso de detecção de amostragem de *frames* de 1,5 segundos, os *frames* serão avaliados em intervalos de tempo mais altos. Portanto, a periodicidade da amostragem de *frames* deve ser selecionada de acordo com as necessidades do administrador, considerando os requisitos específicos do aplicativo e a capacidade do dispositivo com recursos limitados.

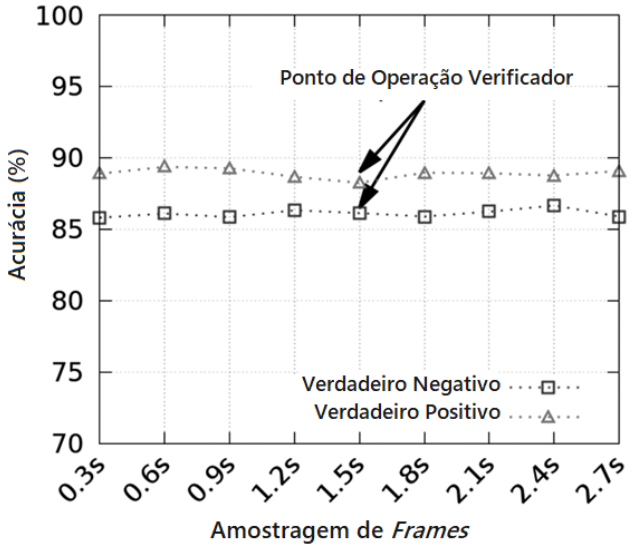

Figura 24 - Acurácia da arquitetura CNN CaffeNet e compensação de amostragem de *frames*

5.5.3. Avaliação da Técnica de Verificação da Confiança da CNN

A presente seção tem como objetivo responder à seguinte RQ7: “A técnica de verificação ajuda na melhoria da acurácia do modelo?”. Essa RQ visa avaliar a melhoria do módulo verificador na acurácia (consulte a Seção 4.4.3). Avaliação realizada na abordagem sobre o módulo de *verificador* em destaque na Figura 13 (Figura 13 - Módulo *verificador*). Para isso, foi utilizada a arquitetura CNN CaffeNet com intervalo de amostragem de *frames* de 1,5 segundo (Disposto na Figura 15, Ponto de operação do verificador). Em seguida, os valores de confiança da CNN foram usados como uma

medida de confiança na classificação de *frames*. Consequentemente, o módulo verificador estabelece se o resultado do frame deve ser aceito ou não, levando em consideração os valores de confiança. Para encontrar os pontos de operação do módulo verificador, foi utilizada a abordagem CRT (*Class Related Thresholds*) (Fumera, et al., 2000) que se baseia em limites específicos de classe para o cálculo da confiança.

Na Figura 25 são mostradas a troca da taxa de erro e taxa de rejeição da arquitetura CNN Caffenet. É possível observar uma relação direta entre as taxas de rejeição e de erro. Em outras palavras, ao aumentar a taxa de rejeição, é possível diminuir a taxa de erro do sistema, melhorando ainda mais a acurácia do sistema. Portanto, o módulo verificador ajuda o sistema de classificação baseado na CNN a aumentar a acurácia obtida, com baixa ou nenhuma demanda de processamento, pois o módulo verificador aplica apenas um limiar simples para a verificação da classificação.

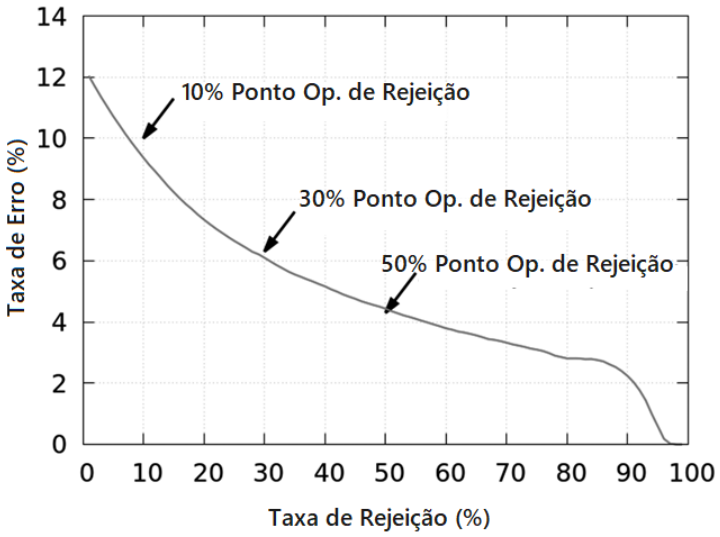

Figura 25 - Troca de rejeição de erro da CNN CaffeNet ao aplicar o módulo verificador com intervalo de 1,5 segundos de amostragem de *frames*

Na Tabela 12 é apresentada com mais detalhes a melhoria da acurácia do módulo verificador aplicando os pontos de operação marcados na Figura 16. É possível observar uma melhoria significativa da acurácia com apenas 10% de taxa de rejeição, melhorando 2,98% e 3,02% em TP e TN, respectivamente. Além disso, ao aumentar ainda mais a taxa de rejeição, por exemplo com uma taxa de rejeição de 30%, as taxas verdadeiro-positiva e verdadeira-negativa aumentam ainda mais em 8,55% e 6,68%, respectivamente.

Tabela 12 - Melhoria da acurácia do CaffeNet através do módulo verificador de confiança proposto

Ponto de Operação	**Acurácia (%)**	***TP (%)***	***TN (%)***
Sem Verificador	88,03	88,99	85,79
10% Rejeição	91,01	91,97	88,81
30% Rejeição	95,93	97,54	92,47
50% Rejeição	97,66	99,42	94,02

Portanto, a técnica de verificação proposta permite melhorar significativamente as precisões de detecção sem trocas significativas no processamento. Um aplicativo que aplica essa técnica é capaz de verificar se o resultado de uma CNN leve pode ser aceito com segurança, diminuindo a proporção de falsos positivos, aumentando ainda mais a confiabilidade do usuário no esquema de detecção.

5.5.4. Avaliação Otimização de Recursos Estado da Arte

Em geral, as arquiteturas CNN no estado da arte visam o aprimoramento da acurácia. Algumas abordagens surgem com a iniciativa da redução do uso de memória ou do aumento de *throughput* na detecção. Contudo, uma arquitetura mais complexa pode exigir mais recursos computacionais para ser processada, muitas vezes inviabilizando o seu uso em dispositivos com recursos limitados.

Por outro lado, devido às limitações de memória dos dispositivos móveis, algumas arquiteturas CNN menores foram propostas. Por exemplo, o Mobilenet (Howard, et al., 2017), proposto pelo Google, adota funções convolucionais em profundidade e separáveis, visando reduzir o número de parâmetros utilizados, tendo como objetivo a redução na demanda por memória. De forma similar, o Squeezenet (Iandola, et al., 2016), também visa diminuir a demanda por memória, diminuindo assim o número de parâmetros necessários para CNN. Por fim, o ShuffleNet (Zhang, et al., 2018) reduz seu tamanho aplicando operações de ponto a ponto convolucional e troca de canais, reduzindo assim as demandas memória e processamento, preservando porém a acurácia.

Observando tais aspectos, foi efetuada uma avaliação empírica entre o desempenho da abordagem proposta e as arquiteturas CNN dipostas no estado da arte, e os resultados dos experimentos seguem dispostos na Tabela 13. Para efetuar a comparação foi selecionado o modelo proposto que obteve o melhor valor na métrica de leveza (Tabela 10). Todas as arquiteturas foram avaliadas seguindo o mesmo protocolo (conforme secção 5.2, Treinamento CNN).

Tabela 13 - Métricas de leveza das técnicas de detecção de conteúdo de vídeo baseadas na CNN

Arquitetura CNN	Acurácia	Taxa de Transferência (FPS)	Memória (MB)
Alexnet	87,56	61,12	217
Caffenet	87,97	63,97	217
GoogleNet	91,51	18,29	48
Mobilenet	88,98	18,53	8,7
PVANet	83,88	10,79	292
Resnet	88,64	22,50	90
Shufflenet	86,76	37,82	3,5
Squeezenet	87,27	31,91	2,9
VGG	91,20	4,26	269
Abordagem Proposta	89,30	301,69	9,0

Nossa técnica melhora significativamente as demandas de memória e a taxa de transferência de detecção, mesmo quando comparadas às arquiteturas CNN projetadas para essa tarefa, com pouco ou nenhum impacto na acurácia. Além disso, somos capazes de melhorar ainda mais a acurácia da detecção aplicando o módulo verificador, e de forma similar aumentamos o *throughput* na detecção aplicando a técnica de amostragem de *frames*.

5.5.5. Avaliação do Stacking Meta Learner com Otimização de Recursos

A abordagem *Stacking Meta Learner* foi projetada para a utilização em dispositivos com recursos limitados. Assim, com o objetivo de reduzir o custo computacional para implementação de nossa abordagem, foram propostas as três técnicas de otimização mencionadas na seção 4.2, avaliação realizada sobre os módulos disposto na Figura 26. Tanto a técnica de amostragem de *frames* como o verificador de confiança são técnicas aplicadas antes de efetivamente adotar a abordagem *Stacking Meta Learner*. Ambas as técnicas são aplicadas de forma reativa, a técnica de amostragem pode ser ativada ou desativada na presença dos *frames* pornográficos, já o módulo verificador de confiança pode ser ativado e desativado dado o valor de confiança da saída da CNN. A terceira técnica propõe uma CNN leve desenvolvida para ser integrada na abordagem *Stacking Meta Learner*. A topologia da arquitetura CNN foi modificada observando as limitações dos dispositivos com recursos limitados, e assim foi proposta a métrica de leveza, ajustada para levar em consideração os requisitos da acurácia, *throughput* (FPS) e memória (Conforme secção 5.5.1).

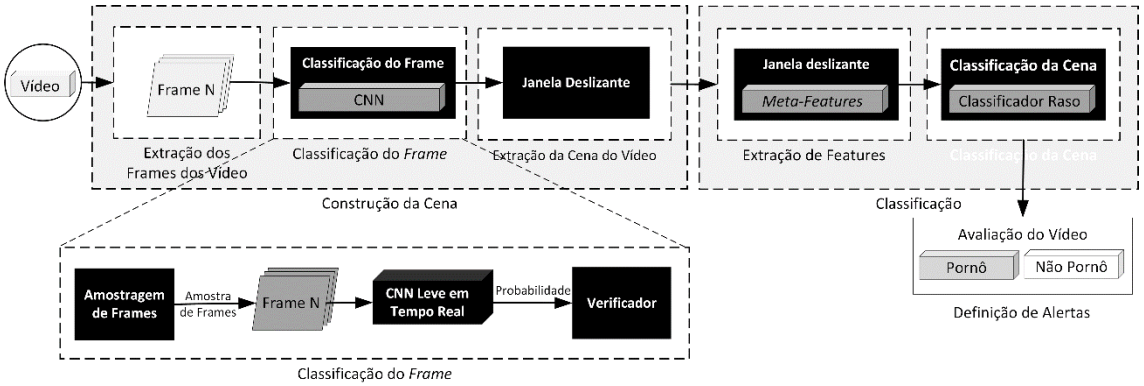

Figura 26 - Avaliação modelo *Stacking Meta Learner* com otimizações de recursos

Desse modo, utilizamos a melhor arquitetura CNN (Tabela 9) mensurada pela métrica de leveza para integrar a abordagem *Stacking Meta Learner*. Para avaliar o impacto da nova arquitetura CNN leve com o *Stacking Meta Learner*, uma série de experimentos foram realizados, onde foi utilizado o *dataset Pornography2k*, e a Tabela 14 dispõe dos resultados.

Tabela 14 - Abordagem *Stacking Meta Learner* com CNN otimizada na métrica de Leveza

Abordagens do Estado da Arte								
Arquiteturas	**Acurácia (%)**	**F1 (%)**	**TP**	**TN**	**FP**	**FN**	***Throughput (FPS)***	**Memória (MB)**
Alexnet	87,56	87,63	87,92	87,40	12,08	12,60	61,12	217
Caffenet	87,97	87,58	86,00	88,82	14,00	11,18	63,97	217
Googlenet	91,51	90,51	86,91	93,49	13,09	6,51	18,29	48
Mobilenet	88,98	91,17	98,85	84,74	1,15	15,26	18,53	8,7
PVANet	83,88	84,41	86,92	82,58	13,08	17,42	10,79	292
Resnet	88,64	87,87	84,76	90,31	15,24	9,69	22,50	90
Shufflenet	86,76	85,47	79,65	89,81	20,35	10,19	37,82	3,5
Squeezenet	87,27	86,01	80,47	90,19	19,53	9,81	31,91	2,9
VGG	91,20	**91,62**	93,04	90,42	6,96	9,58	4,26	269
Abordagens Propostas								
CNN Leve	89,30	87,81	81,75	92,54	18,25	7,46	301,69	9
CNN Leve + Stacking Meta Learner	**91,60**	90,74	87,70	93,27	12,30	6,73	**301,69**	**9 + 3,65KB**

A abordagem proposta apresentou resultados promissores, tornando possível melhorar em até 2,3% de acurácia adotando o *Stacking Meta Learner* sobre o modelo da arquitetura CNN modificada. Destaca-se que o valor da acurácia da abordagem *Stacking Meta Learner* foi o melhor dos modelos propostos na literatura, alcançando uma diferença de 7,72% na acurácia quando comparado com o PVANet.

Contudo, a presente avaliação não está focada em apenas uma métrica, e nosso objetivo é avaliar o melhor conjunto de indicadores destinado a dispositivos com recursos

limitados. Observando que o *dataset Pornography2k* é um *dataset* desbalanceado, a melhor métrica seria utilizar o *F-Measure* (F1), no qual o VGG obteve o melhor resultado com 91,62%. Entretanto, utilizar a arquitetura VGG destinada a um dispositivo com recurso limitado seria a pior opção, pois primeiramente obteve a pior taxa de *throughput* 4,26 FPS (taxa de transferência) e ainda o maior consumo de memória 269MB. No entanto, a abordagem proposta em termos de *F-Measure* é inferior em apenas 0,88% e em contrapartida consegue processar até 301,69 FPS e consumir apenas 9MB aproximadamente.

Por outro lado, se avaliarmos o Squeezenet, sua arquitetura foi a que gerou o menor modelo, consumindo apenas 2,9MB. Por outro lado, ao avaliar sua precisão, o modelo obteve apenas 86,01% em relação ao *F-Measure*, enquanto o *Stacking Meta Learner* alcançou 90,74%. Assim, o Squeezenet acabou sendo 4,73% inferior considerando essa métrica. Contudo, a abordagem *Stacking Meta Learner* com CNN leve, quando comparado com o Squeezenet, não se destaca apenas em relação ao *F-Measure*, observando a taxa de transferência entre os modelos. É possível verificar que o modelo proposto consegue processar cerca de 9,4 vezes mais rápido que o Squeezenet, partindo de 31,91 FPS do Squeezenet para 301,69 FPS da abordagem proposta.

A abordagem *Stacking Meta Learner* utilizando uma *CNN Leve* conseguiu melhorar as taxas de TP e FP quando comparado com a abordagem proposta utilizando apenas o modelo *CNN Leve*, aumetando 5,95% a taxa de TP e reduzindo 5,95% a taxa de FP. Os resultados demostram que a abordagem *Stacking Meta Learner* utilizando uma *CNN Leve* foi superior quando comparada com as arquiteturas CNN do estado da arte, avaliação com foco a utilização em dispositivos com recursos limitados.

5.6. Considerações finais

Os resultados da avaliação demonstraram que a abordagem *Stacking Meta Learner* é eficaz para identificar o contexto dos vídeos, e tal estratégia permitiu um ganho de até 11% em relação abordagem tradicional utilizando CNN (Tabela 6, Google com Transfer Learning). Destaca-se que a abordagem *Stacking Meta Learner* utilizando CNNs com *Transfer Learning* aumentou significativamente a taxa de acurácia na detecção de amostras de *frames Pornô*, aprimorando também a acurácia da detecção das amostras dos *frames Normal* e *Contexto Pornô*. Como resultado, a abordagem proposta também

aumentou significativamente a taxa de TP, tendo um aumento de 26%, em contraste, reduziu a taxa de TN em até 11%, no pior caso.

Um mecanismo de detecção em vídeo baseado em CNN destinado a dispositivos com restrição de recursos deve considerar as vantagens e desvantagens que essa portabilidade introduz. Algumas otimizações nos recursos são indispensáveis. A partir dos resultados foi demostrado que a métrica de leveza da abordagem *Stacking Meta Learner* ajuda na avaliação adequada das compensações introduzidas entre memória, taxa de transferência de detecção e acurácia. Ao aproveitar essa medida, o administrador pode estabelecer adequadamente qual arquitetura CNN deve ser implantada em produção. No entanto, a avaliação da técnica de amostragem de *frames* proposta, a fim de aumentar ainda mais o rendimento da detecção, mostrou que menos *frames* podem ser usados para fins de avaliação e treinamento. Consequentemente, o administrador pode aumentar significativamente o rendimento da detecção, com pouco efeito na acurácia. Finalmente, a avaliação do módulo verificador proposto mostrou que é possível empregar os valores de confiança da CNN como uma medida de confiabilidade para aceitar o resultado da classificação da CNN leve. Portanto, usando limites simples de classificação, o administrador pode melhorar a acurácia de detecção. A avaliação foi eficaz para demostrar que a abordagem *Stacking Meta Learner* permite reduzir o custo computacional, principalmente quando utilizado uma CNN baseada na métrica de leveza.

Comparando as arquiteturas CNN do estado da arte com a abordagem *Stacking Meta Learner*, foi identificado que um aspecto fundamental para seleção da melhor técnica/método é verificar o propósito específico da abordagem. Observando que na seleção da CNN destinada a dispositivos móveis, realizar a seleção da melhor CNN avaliando uma única métrica não seria suficiente, sendo necessário considerar todo cenário. Por fim, os resultados da avaliação mostraram que a arquitetura de detecção de conteúdo de vídeo em tempo real proposta para o processamento de dispositivos com recursos limitados é capaz de melhorar significativamente a eficiência da detecção, acurácia e as demandas de memória.

.

Capítulo 6

Conclusão

As abordagens recorrentes na literatura para detecção de conteúdo pornográficos em vídeos são complexas e exigem alta demanda computacional, e a aplicação de tais abordagens acabam inviabilizadas em dispositivos com limitação de recursos de hardware. Tais abordagens possuem outra limitação, os *datasets* de pornografia publicamente disponíveis que são criados rotulando o vídeo inteiro como pornografia ou não. Contudo, os vídeos pornográficos são compostos por várias cenas (sequência de *frames*), assim pode ser possível que nem todas as cenas do vídeo apresentem conteúdo obsceno ou pornográfico.

Observando tais limitações o presente trabalho foi estruturado. Primeiramente foi proposto um novo *dataset*, utilizado para criar uma técnica de detecção de conteúdo pornográfico em vídeos, esse constituído por *frames* classificados individualmente. Em outras palavras, cada *frame* deve ser avaliado em relação ao seu contexto, independentemente de ser um vídeo com conteúdo pornográfico ou não.

Em geral, as técnicas de detecção de conteúdo obscenos dependem de informações estáticas extraídas de cada *frame* do vídeo, ou da sua combinação, de acordo com o movimento dos *frames* adjacentes. No entanto, a classificação de conteúdo obsceno depende do contexto pois, por exemplo, uma pessoa nua só pode ser classificada como conteúdo pornográfico se o contexto da cena for sexual.

Nossa abordagem é denominada de *Stacking Meta Learner* e propõe um método de classificação estruturado em duas etapas para detecção sensível ao contexto de conteúdo obsceno em vídeos. Primeiro, aplicamos uma rede neural profunda (CNN) para a classificação de quadros individuais. Em seguida, construímos um conjunto de *meta-features*, compreendendo o contexto da cena, e aplicamos um classificador superficial para classificar o contexto da cena. Por meio do novo *dataset* com aproximadamente meio milhão de *frames* de vídeo classificados manualmente, mostramos experimentalmente que as técnicas atuais de detecção falham na detecção de conteúdo pornográfico em

vídeos dependendo do contexto. No entanto, o *Stacking Meta Learner* foi capaz de melhorar significativamente as taxas de classificação ao levar em consideração o contexto do vídeo, confirmando assim a hipótese do trabalho de que a classificação do contexto de vídeo por meio da avaliação de uma janela de *frames* é mais eficaz do que a análise individual que utiliza apenas a saída de uma CNN.

Efetuando uma avaliação na acurácia das abordagens de detecção de última geração (estado da arte), foi observado que as técnicas de detecção de aprendizado profundo utilizadas tradicionalmente são limitadas, incapazes de detectar pornografia em uma cena de vídeos. Além disso, na literatura, a detecção de conteúdo com classificação imprópria em vídeos não é realizada em tempo real. Os resultados da avaliação demonstraram que nossa abordagem proposta foi capaz de melhorar significativamente a detecção de conteúdo com classificação imprópria quase em tempo real nos vídeos, apresentando apenas um *delay* em relação ao tamanho da janela de *frames* avaliada. No entanto, ao usar um conjunto de dados do estado da arte, nossa abordagem obteve resultados semelhantes à literatura, além de poder identificar cenas específicas de conteúdo pornográfico.

Este trabalho ainda abordou o desafio de projetar um esquema de classificação baseado em CNN para dispositivos com recursos limitados. Para cumprir uma tarefa tão desafiadora, foi proposta uma nova arquitetura CNN para otimizar ainda mais a abordagem *Stacking Meta Learner* em dispositivos com recursos limitados. Essa arquitetura leva em consideração as propriedades desejadas de uma abordagem leve na classificação baseada na CNN. Em outras palavras, reunimos os requisitos de memória, a taxa de transferência de detecção e a acurácia durante as tarefas de construção e avaliação da CNN.

Primeiramente, foi introduzido uma métrica de leveza, possibilitando estabelecer adequadamente a melhor CNN para ser utilizada em dispositivos com recursos limitados. Então, empregando nossa técnica de amostragem de *frames*, mostramos que é possível melhorar significativamente o *throughput* da detecção com pouco ou nenhum efeito na acurácia. Finalmente, a proposta de um módulo *verificador* melhora ainda mais a acurácia da nova arquitetura CNN, onde foi aplicado um limiar de classificação simples de acordo com o resultado da classificação. Consequentemente, nossa abordagem proposta melhorou significativamente a acurácia, as demandas de memória e a taxa de transferência de detecção.

Este trabalho teve como resultados parciais até o presente momento uma publicação na *International Joint Conference on Neural Networks* (*IJCNN* 2020), ainda participação como coautor na publicação de *journal* na *Future Generation Computer Systems*. Além de outras duas publicações submetidas, um *magazine* para a *IEEE Intelligent Systems* e outra publicação para *IEEE International Conference on Systems, Man, and Cybernetics* (*SMC* 2020).

Para trabalhos futuros, vamos portar e avaliar o desempenho da abordagem proposta em dispositivos embarcados. Além disso, planejamos avaliar a técnica proposta em outras áreas, como o reconhecimento de objetos de imagem.

Referências Bibliográficas

Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J., 2015. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, *Nature Research*, Volume 33, pp. 831-838.

Andres, R., 2016. *Intel Developer Zone.* [Online] Available at: https://software.intel.com/pt-br/articles/training-and-deploying-deep-learning-networks-with-caffe-optimized-for-intel-architecture [Acesso em 05 Fevereiro 2019].

Anisimov, D. & Khanova, T., 2017. Towards lightweight convolutional neural networks for object detection. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 10, Issue 14.

Ap-apid, R., 2005. An Algorithm for Nudity Detection. *Philippine Computing Science Congress (PCSC)*, 01.

Basilio, M., 2011. Explicit image detection using YCbCr space color model as skin detection. *International Conference Applied Mathematics: on Computer Engineering and Applications*, pp. 123-128.

BRASIL, 2008. Crimes virtuais - Condutas relacionadas à pedofilia na Internet. *Lei n.° 11.829*, 25 Novembro.

Breiman, L., 2001. Random Forests. Volume 45, pp. 5-32.

Burges, C., 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, Volume 2, pp. 121-167.

Caetano, C., Avila, S., Schwartz, W., Guimarães, S. & Araujo, A., 2016. A mid-level video representation based on binary descriptors: a case study for pornography detection. *Neurocomputing*, 12 11, Volume 213, pp. 102-114.

Chen, C.-F. R., Lee, G. G. C., Sritapan, V. & Lin, C.-Y., 2016. Deep Convolutional Neural Network on iOS Mobile Devices. *IEEE International Workshop on Signal Processing Systems*, pp. 130-135.

Ciregan, D., Meier, U. & Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3642-3649.

Coelho, M., 2012. *Conheça os formatos de vídeo mais populares da internet.* [Online] Available at: https://tecnologia.ig.com.br/dicas/internet/2012-06-15/conheca-os-formatos-de-video-mais-populares-da-internet.html [Acesso em 16 Fevereiro 2020].

Collera, V., 2019. *Sim, seus filhos veem pornô (e é assim que isso os afeta).* [Online] Available at: https://brasil.elpais.com/brasil/2019/02/05/eps/1549359489_090898.html [Acesso em 10 Janeiro 2020].

Cybersecurity, W. S., 2019. *Internet Pornography by the Numbers; A Significant Threat to Society.* [Online] Available at: https://www.webroot.com/us/en/resources/tips-articles/internet-pornography-by-the-numbers [Acesso em 10 Janeiro 2020].

Deng, L., Hinton, G. & Kingsbury, B., 2013. New types of deep neural network learning for speech recognition and related applications: An overview. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 26 Maio, pp. 8599-8603.

Deselaers, T., Pimenidis, L. & Ney, H., 2008. Bag-of-visual-words models for adult image classification and filtering. *ICPR - International Conference Pattern Recognition*, pp. 1-4.

Dias, M.S., Oliveira, A.S., Soares, E.N., Amaral, I.R.A & Leite, J.G., 2015. *Da moralidade à patologia: Como a pornografia virtual age no cérebro humano?.* [Online] Available at: https://docplayer.com.br/55016898-Da-moralidade-a-patologia-como-a-pornografia-virtual-age-no-cerebro-humano.html [Acesso em 25 Janeiro 2020].

Domingos, P., 2012. A few useful things to know about machine learning. *Communications of the ACM*, pp. 78-87.

Endeshaw, T., Garcia, J. & Jakobsson, A., 2008. Classification of Indecent Video by Low Complexity Repetitive Motion Detection. *IEEE Applied Imagery Pattern Recognition Workshop*, 15 Outubro.

Enough, 2019. *Enough is Enough Homepage.* [Online] Available at: http://enough.org/stats_porn_industry [Acesso em 25 Janeiro 2020].

Freund, Y. & Schapire, R. E., 1997. A decision-theoretic generalization of online learning and an application to boosting. Volume 55, pp. 119-139.

Fukushima, K., 1980. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, Volume 36, pp. 193-202.

Fumera, G., Roli, F. & Giacinto, G., 2000. Reject option with multiple thresholds. *Pattern Recognition*, Dezembro, Volume 33, pp. 2099-2101.

Gangwar, A., Fidalgo, E., Alegre, E. & González-Castro, V., 2017. Pornography and Child Sexual Abuse Detection in Image and Video: A Comparative Evaluation. *International Conference on Imaging for Crime Detection and Prevention*, 12, pp. 37-42.

Garcia, M.B., Revano, T. F. Jr., Habal, Beau G. M., Contreras, J. O. & Enriquez, J. B. R., 2018. A Pornographic Image and Video Filtering Application Using Optimized Nudity Recognition and Detection Algorithm. *IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management*, 29 Outubro.

Girshick, R., 2015. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440-1448.

Giuseppe, A., Paolo, B., Gabriele, C., Francesco, la T. & Fabio, M., 2009. Detection of images with adult content for parental control on mobile devices?. *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, Setembro, pp. 1-5.

He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778.

Hochreiter, S. & Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, Volume 9, pp. 1735-1780.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Abril.

Huang, L. & Ren, X., 2018. Erotic image recognitionmethod of bagging integrated convolutional neural network. *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, pp. 1-7.

Huang, Y. & Kong, A. W. K., 2016. Using a cnn ensemblefor detecting pornographic and upskirt images. *Biometrics Theory, Applications and Systems (BTAS)*, pp. 1-7.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. & Keutzer, K.V., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. Fevereiro.

Irsoy, O. & Cardie, C., 2014. Opinion mining with deep recurrent neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 720-728.

Jacobson, V., Li, J., Tapia, K. & Morreale, P., 2018. Visualizing Neural Networks for Pattern Recognition. *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*.

Jain, A., Nandakumar, K. & Ross, A., 2005. Score normalization in multimodal biometric systems. *Pattern Recognition*, 12 Dezembro, Volume 38, pp. 2270-2285.

Jansohn, C., Ulges, A. & T. Breuel, T., 2009. Detecting Pornographic Video Content by Combining Image Features with Motion Information. *ACM international conference on Multimedia*, 19 Outubro, pp. 601-604.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. & Darrell, T., 2014. Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675-678.

Ji, S., Xu, W., Yang, M. & Yu, K., 2013. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 32, pp. 221-231.

Jones, M., 2017. *Deep learning architectures.* [Online] Available at: https://www.ibm.com/developerworks/br/library/cc-machine-learning-deep-learning-architectures/index.html [Acesso em 2019 Janeiro 01].

Kaiming, H., Zhang, X., Ren, S. & Sun, J., 2015. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

Karamizadeh, S. & Arabsorkhi, A., 2018. Methods of Pornography Detection: Review. *International Conference on Computer Modeling and Simulation*, 08 Janeiro, pp. 33-38.

Karn, J., 2016. [Online] Available at: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ [Acesso em 10 Fevereiro 2019].

Karpathy, A. & Johnson, J., 2018. *CS231n Convolutional Neural Networks for Visual Recognition.* [Online] Available at: http://cs231n.github.io/ [Acesso em 08 Fevereiro 2019].

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. & Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725-1732.

Kotsiantis, S., Zaharakis, I. & Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. *Emerging Artifial Intelligence Applications in Computer Engineering*, pp. 3-24.

Krizhevsky, A. & Hinton, G., 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I. & Hinton, G., 2012. ImageNet Classification with Deep ConvolutionalNeural Networks. *Neural Information Processing Systems (NIPS)*, pp. 1-9.

Kuroki, Y. N. T. K. S. O. H. a. Y. S., 2007. A psychophysical study of improvements in motion-image quality by using high frame rates.. *Journal of the Society for Information Display,* Volume 15, pp. 61-68.

Lecun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning. *Nature*, Volume 521, pp. 436-444.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-Based Learning Appliend to Document Recognition. *Proceedings of the IEEE*, Volume 86, pp. 2278-2324.

Lee, S., Shim, W. & Kim, S., 2009. Hierarchical system for objectionable video detection. *IEEE Transactions on Consumer Electronics*, pp. 677-684.

Li, Q., Qiu, Z., Yao, T., Mei, T., Rui, Y. & Luo, J., 2016. Action recognition by learning deep multi-granular spatio-temporal video representation. *Proceedings of the 2016 ACM on International Conference on Multimedia*, Junho, pp. 159-166.

Liu, Y., Ouyang, J. & Liu, J., 2017. Bimodal Codebooks Based Adult Video Detection. *Global Conference on Signal and Information Processing*, pp. 1397-1401.

Lopes, A., Avila, S., Peixoto, A., Oliveira, R.S., M. Coelho & Araújo A., 2009. A bag-of-features approach based on Hue-SIFT descriptor for nude detection. *EUSIPCO*, pp. 1552-1556.

Macedo, J., Costa, F. & Santos, J. A. d., 2018. A Benchmark Methodology for Child Pornography Detection. *SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 455-462.

Morais, L. A. d., 2018. *Ciberpedofilia: os crimes de pedofília praticados através da internet.* [Online] Available at: http://www.conteudojuridico.com.br/artigo,ciberpedofilia-os-crimes-de-pedofilia-praticados-atraves-da-internet,590609.html [Acesso em 2019 02 01].

Moreira, D., Avila, S., Perez, M., Moraes, D., Testoni, V., Valle, E., Goldenstein, S. & Rocha, A., 2019. Multimodal data fusion for sensitive scene localization. *Information Fusion*, Janeiro, Volume 45, pp. 307-323.

Moujahid, A., 2016. *A Practical Introduction to Deep Learning with Caffe - Python.* [Online] Available at: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/ [Acesso em 08 Fevereiro 2019].

Nasiri, R.M., Wang, J., Rehman, J., Wang, S. & Wang, Z., 2015. Perceptual quality assessment of high frame rate video. *IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1-6.

Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R. & Toderici, G., 2015. Beyond short snippets: Deep networks for video classification. *IEEE Conference on Computer Vision and Pattern Recognition*, p. 4694–4702.

Nogueira, K., Penatti, O. & Santos, J., 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. Pattern Recognition. *Pattern Recognition*, pp. 539-556.

Nvidia, 2019. *NVIDIA Tegra K1 - Impossibly Advanced.* [Online] Available at: https://www.nvidia.com/object/tegra-k1-processor.html [Acesso em 2020 Janeiro 15].

Parfenovich, D., 2012. *Redes neurais: Da teoria à prática.* [Online] Available at: https://www.mql5.com/ru/articles/497 [Acesso em 09 Fevereiro 2019].

Perez, M., Avilab, S., Moreira, D., Moraesa, D., Testonic, V., Valleb, E., Goldensteina, S. & Rocha, A., 2017. Video Pornography Detection through Deep Learning Techniques Motion Information. *Neurocomputing*, Volume 230, pp. 279-293.

Polastro, M. d. C. & Eleuterio, P. M. d. S., 2012. A statistical approach for identifying videos of child pornography at crime scenes. *International Conference on Availability, Reliability and Security*, pp. 604-612.

Quilan, J. R., 1993. *C4.5: Programs for machine learning.* 1ª ed. San Mateo: Morgan Kaufmann.

Rahman, M. A., Purnama, I. K. E. & Purnomo, M. H., 2014. Simple Method of Human Skin Detection using HSV and YCbCr Color Spaces. *International Conference on Intelligent Autonomous Agents, Networks and Systems*, 19 Agosto, pp. 58-61.

Rea, N., Lacey, G., Lambe, C. & Dahyot, R., 2006. Multimodal periodicity analysis for illicit content detection in videos. *Proceedings of the European Conference on Visual Media Production (CVMP)*, pp. 106-114.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 779-788.

Ren, S., He, K., Girshick, R. & Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), pp. 1137-1149.

Ries, C. X. & Lienhart, R., 2004. A survey on visual adult image recognition. *ACM international conference on Multimedia*, 3 Maio, pp. 661-688.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. & Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, Dezembro, Volume 115, pp. 211-252.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural Networks*, Volume 61, pp. 85-117.

Sokolov, D. & Patkin, M., 2018. Real-time emotion recognition on mobile devices. *IEEE International Conference on Automatic Face & Gesture Recognition*, p. 787.

Soomro, K., Zamir, A. & Shah, M., 2012. UCF101: A *dataset* of 101 human actions classes from videos in the wild.

Steel, C. M., 2012. The Mask-SIFT Cascading Classifier for Pornography Detection. *World Congress on Internet Security*, pp. 139-142.

Storey, D., 2018. *Random Forests, Decision Trees, Ensemble Methods Explained.* [Online] Available at: https://www.datascience.com/blog/random-forests-decision-trees-ensemble-methods [Acesso em 10 Fevereiro 2019].

Strigl, D., Kofler, K. & Podlipnig, S., 2010. Performance; Scalability of GPU-Based Convolutional Neural Networks. *Euromicro Conference on Parallel, Distributed and Network-based Processing*, pp. 317-324.

Sutskever, I., Martens, J., Dahl, G. & Hinton, G., 2013. On the importance of initialization; momentum in deep learning. *Mach Learn*, Volume 23, pp. 1139-1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A., 2015. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7-12.

Ulges, A., Schulze, C., Borth, D. & Stahl, A., 2012. Pornography detection in video benefits (a lot) from a multi-modal approach. *Proceedings of the 2012 ACM international workshop on Audio and multimedia methods for large-scale video analysis*, pp. 21-26.

Wang, J., Cao, B., Yu, P.S., Sun, L., Bao, W. & Zhu, X., 2018. Deep Learning Towards Mobile Applications. *International Conference on Distributed Computing Systems*, pp. 1385-1393.

Wedemann, K., 2019. *O que a Inteligência Artificial significa para seu negócio?*. [Online] Available at: https://www.sas.com/pt_br/insights/articles/analytics/o-que-inteligencia-artificial-significa-para-seu-negocio.html [Acesso em 15 fevereiro 2019].

Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., Zhao, Y. & Yan, S., 2016. HCP: A Flexible CNN Framework for Multi-Label Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 Outubro, pp. 1901-1907.

Weka, 2019. *Weka 3: Data Mining Software in Java - Class J48.* [Online] Available at: http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html [Acesso em 01 Fevereiro 2020].

Wiratama, M. R., Endah, S. N., Kusumaningrum, R. & Wibawa, H. A., 2017. Pornography Object Detection Using Viola-Jones Algorithm and Skin Detection. *International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 29-34.

Wu, C., Zhang, L., Li, Q., Fu, Z., Zhu, W. & Zhang, Y., 2019. Enabling Flexible Resource Allocation in Mobile Deep Learning Systems. *IEEE Transactions On Parallel and Distributed Systems*, 02, pp. 346-360.

Wu, Z., Wang, X., Jiang, Y., Ye, H. & Xue, X., 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 461-470.

Yagielowicz, S., 2012. *The Internet Really is Porn.* [Online] Available at: http://www.xbiz.com/news/146703 [Acesso em 16 Janeiro 2019].

Yu, J.-J. & Han, S.-W., 2012. Skin Detection for Adult Image Identification. *IEEE/ICACT - International Conference of Advanced Communications Technology*, 16 02, pp. 645-648.

Zhang, H., 2004. The optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*.

Zhang, X., Zhou, X., Lin, M. & Sun, J., 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dezembro.pp. 6848-6856.

Zha, S., Luisier, F., Andrews, W., Srivastava, N. & Salakhutdinov, R., 2015. Exploiting image-trained cnn architectures for unconstrained video classification. *International Conference on Learning Representations*.

Zhu, H., Zhou, S., Wang, J. & Yin, Z., 2007. An algorithm of pornographic image detection. *ICIG - International Conference Image and Graphics*, pp. 801-804.

Apêndices

Apêndice A - Seleção de Indutores

Este capítulo apresenta na íntegra os resultados obtidos dos experimentos realizados para selecionar os indutores utilizados no projeto. Os experimentos foram realizados utilizando a ferramenta WEKA (dispostos na secção 5.1.1) e diferentes algoritmos de aprendizagem de máquina foram testados. Os classificadores no WEKA recebem como entrada as características extraídas da abordagem *Stacking Meta Learner*, absorvidas da janela deslizante de *frames*, posteriormente estruturadas em um arquivo "*.arff"*.

A.1. Seleção Classificadores *Dataset* Proposto

A Tabela A1 do apêndice dispõem dos resultados dos classificadores que foram executados no WEKA. Essa seção corresponde aos experimentos realizados sobre o *dataset* proposto. Nesses experimentos a abordagem *Stacking Meta Learner* utiliza os modelos treinados de maneira convencional (sem *Transfer Learning*), correspondente a cada uma das arquiteturas CNN selecionadas (Alexnet, Caffenet e Googlenet).

Tabela A1 – Seleção indutores – arquiteturas CNN tradicional / *dataset* proposto

tipo	***Classifiers***	***Dataset Stacking Meta Learner (Dataset Proposto)***					
		Alexnet		***Caffenet***		***Googlenet***	
		ACC	***F1***	***ACC***	***F1***	***ACC***	***F1***
bayes	***BayesNet***	83,4568	0,834	81,6201	0,817	83,5202	0,835
	NaiveBayes	83,5518	0,835	82,2028	0,824	82,1268	0,822
	NaiveBayesMultinomial	83,3492	0,833	81,3541	0,817	81,7088	0,817
	NaiveBayesMultinomialText	66,6667	0,800	66,6667	0,800	66,6667	0,800
	NaiveBayesMultinomialUpdateable	83,3492	0,833	81,3541	0,817	81,7088	0,817
	NaiveBayesUpdateable	83,5518	0,835	82,2028	0,824	82,1268	0,822
functions	***LibSVM***	79,3970	0,790	77,6173	0,777	78,5610	0,785
	Logistic	79,1690	0,788	77,5033	0,776	74,1782	0,743
	MultilayerPerceptron	79,5871	0,793	77,5160	0,776	78,3710	0,781
	SGD	77,9277	0,777	76,3190	0,765	76,4520	0,763
	SGDText	66,6667	0,800	66,6667	0,800	66,6667	0,800

	SimpleLogistic	77,9277	0,777	76,3190	0,765	78,2950	0,780
	SMO	77,9277	0,777	76,3190	0,765	76,3760	0,762
	VotedPerceptron	80,0114	0,797	77,5287	0,776	79,3400	0,792
lazy	***Ibk***	78,4407	0,783	76,3506	0,766	81,4491	0,811
	Kstar	82,8868	0,824	80,9488	0,809	82,6968	0,823
	LWL	77,9277	0,777	76,3190	0,765	76,3760	0,762
meta	***AdaBoostM1***	77,9277	0,777	76,3190	0,765	77,4780	0,772
	AttributeSelectedClassifier	77,9277	0,777	76,3190	0,765	77,2943	0,774
	Bagging	78,3584	0,782	76,3190	0,765	77,9213	0,779
	ClassificationViaRegression	78,4217	0,782	76,3190	0,765	78,7257	0,787
	CostSensitiveClassifier	66,6667	0,800	66,6667	0,800	66,6667	0,800
	CVParameterSelecion	66,6667	0,800	66,6667	0,800	66,6667	0,800
	FilteredClassifier	77,9277	0,777	76,3190	0,765	77,2943	0,774
	IterativeClassifierOptimizer	77,9277	0,777	76,3190	0,765	78,2887	0,783
	LogiBoost	77,9277	0,777	76,3190	0,765	78,2887	0,783
	MultiClassClassifier	79,1690	0,788	77,5033	0,776	74,1782	0,743
	MultiClassClassifierUpdateable	77,9277	0,779	76,3190	0,765	76,4520	0,763
	MultiScheme	66,6667	0,800	66,6667	0,800	66,6667	0,800
	RandomCommittee	78,3203	0,782	76,3506	0,766	78,0543	0,780
	RandomizableFilteredClassifier	86,1169	0,853	70,1944	0,709	80,8601	0,806
	RandomSubSpace	85,1859	0,846	76,3190	0,765	81,0501	0,808
	Stacking	66,6667	0,800	66,6667	0,800	66,6667	0,800
	Vote	66,6667	0,800	66,6667	0,800	66,6667	0,800
	WeightedInstancesHandlerWrapper	66,6667	0,800	66,6667	0,800	66,6667	0,800
misc	***InputMappedClassifier***	66,6667	0,800	66,6667	0,800	66,6667	0,800
	SerializedClassifier (NaiveBayes)	83,5518	0,835	82,2028	0,824	82,1268	0,822
rules	***DecisionTable***	81,5124	0,810	76,3190	0,765	81,0374	0,806
	JRip	78,4027	0,782	76,3190	0,765	78,3837	0,784
	OneR	77,9277	0,777	76,3190	0,765	76,3760	0,762
	PART	78,1937	0,780	76,3190	0,765	80,2837	0,801
	ZeroR	66,6667	0,800	66,6667	0,800	66,6667	0,800
trees	***DecisionStump***	77,9277	0,777	76,3190	0,765	76,3760	0,762
	HoeffdingTree	75,6223	0,756	72,4492	0,730	77,0283	0,771
	J48	78,1937	0,780	76,3190	0,765	79,9861	0,799
	LMT	78,7954	0,785	76,3190	0,765	78,4977	0,785
	RandomForest	78,5230	0,784	76,3506	0,766	78,7194	0,786
	RandomTree	78,6814	0,785	76,3506	0,766	78,7320	0,786
	REPTree	78,2507	0,781	76,3190	0,765	77,5033	0,774

Legenda:
ACC – Acurácia
F1 – F-Measure ou F1 Score

A.2. Seleção Classificadores *Dataset* Proposto Com Transfer Learning

A Tabela A2 do apêndice dispõem dos resultados dos classificadores que foram executados no WEKA. Essa seção corresponde aos experimentos realizados sobre o *dataset* proposto. Os experimentos dispostos nessa seção são referentes a abordagem *Stacking Meta Learner* utilizando os modelos treinados com *Transfer Learning* aplicados sobre as arquiteturas CNN selecionadas (Alexnet, Caffenet e Googlenet).

Tabela A2 – Seleção indutores – arquiteturas CNN com *Transfer Learning* / *dataset* proposto

tipo	***Classifiers***	***Dataset Stacking Meta Learner (Dateset Proposto)***					
		Transfer Alexnet		***Transfer Caffenet***		***Transfer Googlenet***	
		ACC	***F1***	***ACC***	***F1***	***ACC***	***F1***
bayes	***BayesNet***	84,1979	0,843	88,3337	0,884	88,7073	0,889
	NaiveBayes	83,5708	0,838	87,5990	0,878	85,8636	0,862
	NaiveBayesMultinomial	83,1655	0,834	87,1049	0,873	85,4139	0,858
	NaiveBayesMultinomialText	66,6667	0,800	66,6667	0,800	66,6667	0,800
	NaiveBayesMultinomialUpdateable	83,1655	0,834	87,1049	0,873	85,4139	0,858
	NaiveBayesUpdateable	83,5708	0,838	87,5990	0,878	85,8636	0,862
functions	***LibSVM***	79,9861	0,801	84,1409	0,843	83,6595	0,841
	Logistic	77,5413	0,779	80,9361	0,812	83,6025	0,840
	MultilayerPerceptron	80,0937	0,802	84,1662	0,843	83,6532	0,841
	SGD	78,5674	0,788	82,7855	0,830	82,0761	0,825
	SGDText	66,6667	0,800	66,6667	0,800	66,6667	0,800
	SimpleLogistic	78,5674	0,788	84,1092	0,842	83,5518	0,840
	SMO	78,5674	0,788	82,7855	0,830	81,9938	0,825
	VotedPerceptron	80,0304	0,802	84,1915	0,843	83,5708	0,840
lazy	***Ibk***	82,5385	0,825	86,3956	0,864	87,0163	0,872
	Kstar	83,6722	0,836	88,4730	0,884	88,7707	0,890
	LWL	78,5674	0,788	82,7855	0,830	81,9938	0,825
meta	***AdaBoostM1***	78,5674	0,788	83,2605	0,834	83,0895	0,835
	AttributeSelectedClassifier	78,5674	0,788	82,7855	0,830	82,3105	0,828
	Bagging	78,5674	0,788	83,3365	0,835	83,3492	0,838
	ClassificationViaRegression	79,2450	0,794	83,0768	0,833	83,1845	0,836
	CostSensitiveClassifier	66,6667	0,800	66,6667	0,800	66,6667	0,800
	CVParameterSelecion	66,6667	0,800	66,6667	0,800	66,6667	0,800
	FilteredClassifier	78,5674	0,788	82,7855	0,830	82,4245	0,829
	IterativeClassifierOptimizer	78,5674	0,788	83,4948	0,836	83,3302	0,837
	LogiBoost	79,2767	0,795	84,1282	0,842	83,3302	0,837
	MultiClassClassifier	77,5413	0,779	80,9361	0,812	83,6025	0,840
	MultiClassClassifierUpdateable	78,5674	0,788	82,7855	0,830	82,0761	0,825
	MultiScheme	66,6667	0,800	66,6667	0,800	66,6667	0,800
	RandomCommittee	79,0297	0,792	83,2985	0,834	83,9002	0,843
	RandomizableFilteredClassifier	74,2669	0,748	86,2563	0,862	80,2837	0,808
	RandomSubSpace	78,5674	0,788	85,1416	0,852	83,8875	0,843

	Stacking	66,6667	0,800	66,6667	0,800	66,6667	0,800
	Vote	66,6667	0,800	66,6667	0,800	66,6667	0,800
	WeightedInstancesHandlerWrapper	66,6667	0,800	66,6667	0,800	66,6667	0,800
misc	***InputMappedClassifier***	66,6667	0,800	66,6667	0,800	66,6667	0,800
	SerializedClassifier (NaiveBayes)	83,5708	0,838	87,5990	0,878	85,8636	0,862
rules	***DecisionTable***	82,2851	0,823	86,1803	0,862	86,4083	0,867
	JRip	78,5800	0,788	82,7855	0,830	83,3048	0,837
	OneR	78,5674	0,788	82,7855	0,830	81,9938	0,825
	PART	78,9284	0,791	83,6278	0,838	83,2162	0,836
	ZeroR	66,6667	0,800	66,6667	0,800	66,6667	0,800
trees	***DecisionStump***	78,5674	0,788	82,7855	0,830	81,9938	0,825
	HoeffdingTree	77,6807	0,780	82,7918	0,830	83,1465	0,836
	J48	78,5674	0,788	82,7855	0,830	83,0958	0,835
	LMT	78,5674	0,788	84,1092	0,842	83,5518	0,840
	RandomForest	79,1184	0,793	83,6215	0,838	83,7798	0,842
	RandomTree	79,2070	0,794	84,2042	0,843	83,9635	0,843
	REPTree	78,5674	0,788	82,7348	0,829	83,2542	0,837
Legenda: *ACC – Acurácia* *F1 – F-Measure ou F1 Score*							

A.3. Seleção Classificadores *Dataset Pornography2k*

A Tabela A3 do apêndice dispõem dos resultados dos classificadores que foram executados no WEKA. Essa seção corresponde aos experimentos realizados sobre o *dataset Pornography2k*. Nesses experimentos a abordagem *Stacking Meta Learner* utiliza os modelos treinados de maneira convencional (sem *Transfer Learning*), correspondentes a cada uma das arquiteturas CNN selecionadas (Alexnet, Caffenet e Googlenet).

Tabela A3 – Seleção indutores – arquiteturas CNN tradicional / *dataset Pornography2k*

tipo	***Classifiers***	***Dataset Pornography2k***					
		Alexnet		***Caffenet***		***Googlenet***	
		ACC	***F1***	***ACC***	***F1***	***ACC***	***F1***
bayes	***BayesNet***	89,1949	0,894	89,0802	0,892	92,4479	0,925
	NaiveBayes	89,359	0,895	89,2504	0,894	92,6185	0,927
	NaiveBayesMultinomial	89,452	0,896	89,4322	0,895	92,4479	0,924
	NaiveBayesMultinomialText	69,9597	0,823	69,9597	0,823	69,9597	0,823
	NaiveBayesMultinomialUpdateable	89,452	0,896	89,4322	0,895	92,4479	0,924
	NaiveBayesUpdateable	89,359	0,895	89,2504	0,894	92,6185	0,927
functions	***LibSVM***	89,8596	0,900	89,8694	0,900	92,9633	0,930
	Logistic	89,8423	0,900	89,9505	0,901	93,0232	0,931
	MultilayerPerceptron	89,8878	0,900	89,9303	0,900	93,0077	0,931
	SGD	89,9144	0,900	89,8878	0,900	92,9597	0,930
	SGDText	69,9597	0,823	69,9597	0,823	69,9597	0,823
	SimpleLogistic	89,8474	0,900	89,9303	0,900	93,0376	0,931

	SMO	89,8903	0,900	89,9444	0,901	92,9799	0,930
	VotedPerceptron	89,6710	0,898	89,3215	0,895	92,8010	0,929
lazy	***Ibk***	89,7359	0,899	89,7947	0,899	92,8771	0,929
	Kstar	89,8546	0,900	89,9188	0,900	92,9908	0,930
	LWL	89,8683	0,900	89,3961	0,896	93,0124	0,931
meta	***AdaBoostM1***	89,7669	0,899	89,6573	0,898	93,0214	0,930
	AttributeSelectedClassifier	89,2194	0,893	88,9792	0,891	92,9525	0,930
	Bagging	89,9090	0,900	89,9094	0,900	92,9514	0,930
	ClassificationViaRegression	89,8726	0,900	89,9621	0,901	92,9691	0,930
	CostSensitiveClassifier	69,9597	0,823	89,2504	0,894	69,9597	0,823
	CVParameterSelecion	69,9597	0,823	69,9597	0,823	69,9597	0,823
	FilteredClassifier	89,8708	0,900	89,8968	0,900	92,9565	0,930
	IterativeClassifierOptimizer	89,7669	0,899	89,8834	0,900	93,0214	0,930
	LogiBoost	89,7669	0,899	89,8834	0,900	93,0214	0,930
	MultiClassClassifier	89,8423	0,900	89,9505	0,901	93,0232	0,931
	MultiClassClassifierUpdateable	89,9144	0,900	89,8878	0,900	92,9597	0,930
	MultiScheme	69,9597	0,823	69,9597	0,823	69,9597	0,823
	RandomCommittee	89,7428	0,899	89,7878	0,899	92,8822	0,929
	RandomizableFilteredClassifier	89,7518	0,899	89,7878	0,899	92,8678	0,929
	RandomSubSpace	89,8452	0,900	89,9130	0,900	92,9587	0,930
	Stacking	69,9597	0,823	69,9597	0,823	69,9597	0,823
	Vote	69,9597	0,823	69,9597	0,823	69,9597	0,823
	WeightedInstancesHandlerWrapper	69,9597	0,823	69,9597	0,823	69,9597	0,823
misc	***InputMappedClassifier***	69,9597	0,823	69,9597	0,823	69,9597	0,823
	SerializedClassifier (NaiveBayes)	89,3590	0,895	89,2504	0,894	92,6185	0,927
rules	***DecisionTable***	89,9112	0,900	89,9512	0,901	92,9684	0,930
	JRip	89,8755	0,900	89,9090	0,900	92,9623	0,930
	OneR	89,7669	0,899	89,8834	0,900	93,0214	0,930
	PART	89,8167	0,900	89,9094	0,900	92,9457	0,930
	ZeroR	69,9597	0,823	69,9597	0,823	69,9597	0,823
trees	***DecisionStump***	89,7669	0,899	89,6573	0,898	93,0214	0,930
	HoeffdingTree	69,9597	0,823	69,9597	0,823	69,9597	0,823
	J48	89,8964	0,900	89,9321	0,900	92,9460	0,930
	LMT	89,9155	0,900	89,9707	0,901	92,9727	0,930
	RandomForest	89,8243	0,900	89,8510	0,900	92,9085	0,930
	RandomTree	89,7345	0,899	89,7857	0,899	92,8721	0,929
	REPTree	89,8939	0,900	89,8924	0,900	92,9190	0,930

Legenda:
ACC – Acurácia
F1 – F-Measure ou F1 Score

A.4. Seleção Classificadores *Dataset Pornography2k com Transfer Learnig*

A Tabela A4 do apêndice dispõem dos resultados dos classificadores que foram executados no WEKA. Essa seção corresponde aos experimentos realizados sobre o

dataset Pornography2k. Os experimentos dispostos nessa seção são referentes à abordagem *Stacking Meta Learner* utilizando os modelos treinados com *Transfer Learning* aplicados sobre as arquiteturas CNN selecionadas (Alexnet, Caffenet e Googlenet).

Tabela A4 – Seleção indutores – arquiteturas CNN com *Transfer Learning* / *dataset Pornography2k*

tipo	***Classifiers***	***Dataset Pornography2k***					
		Transfer Alexnet		***Transfer Caffenet***		***Transfer Googlenet***	
		ACC	***F1***	***ACC***	***F1***	***ACC***	***F1***
bayes	***BayesNet***	91,4686	0,915	93,9877	0,940	96,2983	0,963
	NaiveBayes	91,4676	0,915	94,2827	0,943	96,3271	0,963
	NaiveBayesMultinomial	91,0481	0,909	93,9404	0,939	96,1803	0,962
	NaiveBayesMultinomialText	69,9597	0,823	69,9597	0,823	69,9597	0,823
	NaiveBayesMultinomialUpdateable	91,0480	0,909	93,9404	0,939	96,1803	0,962
	NaiveBayesUpdateable	91,9433	0,920	94,2827	0,943	96,3271	0,963
Functions	***LibSVM***	92,0678	0,921	94,6676	0,947	96,3892	0,964
	Logistic	91,9938	0,920	94,6982	0,947	96,4425	0,964
	MultilayerPerceptron	91,9246	0,919	94,7080	0,947	96,3167	0,963
	SGD	91,9112	0,919	94,6568	0,947	96,4151	0,964
	SGDText	69,9597	0,823	69,9597	0,823	69,9597	0,823
	SimpleLogistic	91,9960	0,920	94,6878	0,947	96,4396	0,964
	SMO	91,9018	0,919	94,6734	0,947	96,4371	0,964
	VotedPerceptron	91,8636	0,910	94,3397	0,944	96,2543	0,962
Lazy	***Ibk***	91,9581	0,920	94,6265	0,946	96,3841	0,964
	Kstar	91,9963	0,920	94,6900	0,947	96,4267	0,964
	LWL	92,0277	0,920	94,7037	0,947	92,9085	0,964
Meta	***AdaBoostM1***	92,0371	0,920	94,6564	0,947	96,4396	0,964
	AttributeSelectedClassifier	91,9927	0,920	94,7044	0,947	96,3700	0,963
	Bagging	92,0252	0,920	94,7047	0,947	96,3700	0,964
	ClassificationViaRegression	91,9870	0,920	94,7156	0,947	96,3610	0,964
	CostSensitiveClassifier	69,9597	0,823	69,9597	0,823	69,9597	0,823
	CVParameterSelecion	69,9597	0,823	69,9597	0,823	69,9597	0,823
	FilteredClassifier	91,9934	0,920	94,7073	0,947	96,3630	0,963
	IterativeClassifierOptimizer	92,0219	0,920	94,6564	0,947	96,4396	0,964
	LogiBoost	92,0219	0,920	94,6564	0,947	96,4396	0,964
	MultiClassClassifier	91,9938	0,920	94,6982	0,947	96,4425	0,964
	MultiClassClassifierUpdateable	91,9112	0,919	94,6568	0,947	96,4151	0,964
	MultiScheme	69,9597	0,823	69,9597	0,823	69,9597	0,823
	RandomCommittee	91,9401	0,919	94,6456	0,947	96,3888	0,964
	RandomizableFilteredClassifier	91,9520	0,920	94,6290	0,946	96,3877	0,964
	RandomSubSpace	91,9989	0,920	94,6611	0,947	96,3625	0,963
	Stacking	69,9597	0,823	69,9597	0,823	69,9597	0,823
	Vote	69,9597	0,823	69,9597	0,823	69,9597	0,823
	WeightedInstancesHandlerWrapper	69,9597	0,823	69,9597	0,823	69,9597	0,823

Misc	***InputMappedClassifier***	69,9597	0,823	69,9597	0,823	69,9597	0,823
	SerializedClassifier (NaiveBayes)	91,7983	0,919	94,2827	0,943	96,3271	0,963
Rules	***DecisionTable***	92,0407	0,920	94,7037	0,947	96,3397	0,963
	JRip	92,0598	0,921	94,7109	0,947	96,3708	0,964
	OneR	92,0371	0,920	94,6564	0,947	96,4396	0,964
	PART	91,9642	0,919	94,6932	0,947	96,3697	0,964
	ZeroR	69,9597	0,823	69,9597	0,823	69,9597	0,823
Trees	***DecisionStump***	92,0371	0,920	94,6564	0,947	96,4396	0,964
	HoeffdingTree	69,9597	0,823	69,9597	0,823	69,9597	0,823
	J48	91,9906	0,920	94,7087	0,947	96,3700	0,963
	LMT	92,0252	0,920	94,7098	0,947	96,3747	0,964
	RandomForest	91,9498	0,919	94,6669	0,947	96,3892	0,964
	RandomTree	91,9238	0,919	94,6355	0,946	96,3715	0,964
	REPTree	91,9884	0,920	94,7105	0,947	96,3744	0,964

Legenda:
ACC – Acurácia
F1 – F-Measure ou F1 Score

Apêndice B - Topologia CNN

Este capítulo apresenta na íntegra os resultados obtidos dos experimentos realizados para modificar a topologia da arquitetura CNN. Os experimentos foram realizados tendo por base a arquitetura CNN Caffenet (dispostos na secção 5.5.1), diferentes disposições nas camadas da arquitetura CNN foram testadas – remoção e adição das camadas intermediárias. Experimento realizado para verificar o impacto da arquitetura em relação ao tamanho do modelo e na acurácia.

B.1. Topologia da Arquitetura CNN Caffenet

A Tabela B1 é um esquema que dispõem do grupo de camadas utilizado para gerar cada um dos modelos modificados do Caffenet.

Tabela B1 – Estrutura da topologia da arquiteturas CNN Caffenet para modificação

ways	*conv1*	*relu1*	*pool1*	*norm1*	*conv2*	*relu2*	*pool2*	*norm2*	*conv3*	*relu3*	*conv4*	*relu4*	*conv5*	*relu5*	*pool5*	*fc6 (4096)*	*relu6*	*drop6*	*fc7 (4096)*	*relu7*	*drop7*	*fc8 (2)*	*loss*	*acurracy*
step	1				2			3			4		5			6			7			8		
full	X				X			X			X		X			X			X			X		
1,3,4,5,6,7 e 8	X				-			X			X		X			X			X			X		
1,4,5,6,7 e 8	X				-			-			X		X			X			X			X		
1,5,6,7 e 8	X				-			-			-		X			X			X			X		
1,6,7 e 8	X				-			-			-		-			X			X			X		
1,7 e 8	X				-			-			-		-			-			X			X		
1 e 8	X				-			-			-		-			-			-			X		
1,2,3,4,5,6 e 8	X				X			X			X		X			X			-			X		
1,2,3,4,5 e 8	X				X			X			X		X			-			-			X		
1,2,3,4 e 8	X				X			X			X		-			-			-			X		
1,2,3 e 8	X				X			X			-		-			-			-			X		
1,2 e 8	X				X			-			-		-			-			-			X		
1,2,5 e 8	X				X			-			-		X			-			-			X		

B.2. Experimentos Topologia Caffenet - GPU Titan XP/ CPU

A Tabela B2 apresenta os resultados integrais dos experimentos realizados na GPU Titan XP em paralelo com os resultados da CPU. Experimentos realizados sobre o *dataset Pornograpy2k,* observando o tempo necessário para classificar todo o conjunto de teste. Os dados inferem que determinadas camadas influenciam diretamente no tamanho do modelo, bem como na acurácia obtida.

Tabela B2 – Experimentos de impacto modificação arquitetura CNN Caffenet - GPU Titan XP / CPU

Ways	***Size***	***Accuracy***	***Time Real***	***Time User***	***Sys***	***Time Teal***	***Time User***	***Sys***
Step			***GPU Titan XP***			***CPU***		
Full	217MB	88,02733	98m53.494s	36m8.666s	3m39.023s	863m33.377s	765m51.449s	1m49.357s
1,3,4,5,6,7 e 8	746MB	89,96855	136m9.003s	41m37.827s	6m32.564s	1828m1.523s	1734m52.237s	1m59.289s
1,4,5,6,7 e 8	743MB	88,91714	129m46.226s	44m51.812s	7m4.101s	1743m2.549s	1611m12.268s	2m10.594s
1,5,6,7 e 8	741MB	89,16205	98m41.021s	43m18.801s	6m53.826s	1595m53.153s	1464m59.649s	2m7.331s
1,6,7 e 8	1,2GB	84,50701	109m57.775s	49m26.650s	9m22.111s	2336m53.672s	2205m49.312s	2m13.131s
1,7 e 8	1,1GB	84,47888	111m44.076s	48m39.973s	9m0.549s	2233m38.096s	2102m46.041s	2m13.747s
1 e 8	684KB	73,63804	114m33.402s	32m59.930s	2m58.291s	349m15.541s	110m19.398s	1m49.216s
1,2,3,4,5,6 e 8	153MB	89,47441	121m21.127s	35m35.463s	3m28.960s	781m28.701s	661m12.349s	1m56.866s
1,2,3,4,5 e 8	9,0MB	89,30272	142m1.009s	33m34.952s	2m48.246s	512m25.483s	395m6.391s	1m49.320s
1,2,3,4 e 8	7,8MB	88,32634	155m40.031s	35m11.636s	3m37.976s	474m44.755s	350m55.552s	1m54.367s
1,2,3 e 8	5,2MB	88,37431	102m27.114s	34m48.776s	3m16.936s	454m0.288s	332m59.439s	1m51.658s
1,2 e 8	1,7MB	87,07836	154m37.842s	33m41.977s	3m6.538s	234m31.856s	202m47.819s	1m21.560s
1,2,5 e 8	2,6MB	87,42462	148m36.268s	31m49.370s	2m21.006s	390m31.098s	262m17.552s	1m53.832s

B.3. Experimentos Topologia Caffenet - TK1 GPU (GK20a) / CPU

A Tabela B3 apresenta os resultados integrais dos experimentos realizados na placa de desenvolvimento da NVDIA Jetson TK1, utilizado a GPU GK20 em paralelo com os resultados realizado apenas com a CPU. Experimentos realizados sobre o *dataset Pornograpy2k,* observando o tempo necessário para classificar todo o conjunto de teste. Os dados inferem que determinadas camadas influenciam diretamente no tamanho do modelo, bem como na acurácia obtida.

Tabela B3 – Experimentos de impacto modificação arquitetura CNN Caffenet - TK1 GPU (GK20A) / CPU

Ways	***Size***	***Accuracy***	***Time Teal***	***Time User***	***Sys***	***Time Real***	***Time User***	***Sys***
Step			***TK1 GPU (GK20A)***			***CPU TK1***		
Full	217MB	88,0273	534m57.975s	182m7.771s	64m29.891s	3763m15.177s	3542m45.354s	18m47.580s
1,3,4,5,6,7 e 8	746MB	89,9685	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**
1,4,5,6,7 e 8	743MB	88,9171	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**
1,5,6,7 e 8	741MB	89,1620	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**

1,6,7 e 8	1,2GB	84,5070	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**
1,7 e 8	1,1GB	84,4789	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**	*Exception**
1 e 8	684KB	73,6380	331m47.448s	124m57.062	21m40.052s	673m31.177s	567m58.045s	14m20.902s
1,2,3,4,5,6 e 8	153MB	89,4744	563m49.390s	179m24.638s	36m29.439s	3275m32.686s	3071m8.310s	18m54.380s
1,2,3,4,5 e 8	9,0MB	89,3027	420m42.590s	144m34.325s	28m1.1572	2410m17.557s	2238m17.969s	17m28.701s
1,2,3,4 e 8	7,8MB	88,3263	451m37.664s	152m41.546s	29m31.359s	2181m39.851s	2008m49.171s	18m11.751s
1,2,3 e 8	5,2MB	88,3743	418m28.984s	144m35.087s	27m0.801s	2044m25.446s	1879m1.567s	16m57.302s
1,2 e 8	1,7MB	87,0784	375m7.981s	134m38.606s	23m42.979s	1384m30.661s	1236m4.887s	16m7.887s
1,2,5 e 8	2,6MB	87,4246	391m13.799s	136m32.436s	26m2.464s	1646m1.890s	1505m9.366s	15m28.662s
Legenda: *Exception* - Limit of Memory*								

B.4. Experimentos Topologia Caffenet – GPU GeForce GTX 750 Ti / CPU

A Tabela B4 apresenta os resultados integrais dos experimentos realizados na GPU GeForce GTX 750 Ti em paralelo com os resultados da CPU. Experimentos realizados sobre o *dataset Pornograpy2k*, observando o tempo necessário para classificar todo o conjunto de teste. Os dados inferem que determinadas camadas influenciam diretamente no tamanho do modelo, bem como na acurácia obtida.

Tabela B4 – Experimentos impacto modificação arquitetura CNN Caffenet - GPU Geforce GTX 750 Ti / CPU

Ways	***Size***	***Accuracy***	***Time Real***	***Time User***	***Sys***	***Time Real***	***Time User***	***Sys***
Step			***GeForce GTX 750 Ti - GPU***			***CPU***		
Full	217MB	88,02732572	123m58.898s	48m25.572s	9m51.412s	883m6.550s	784m41.588s	1m27.788s
1,3,4,5,6,7 e 8	746MB	89,96854801	157m33.645s	76m25.288s	22m41.512s	1729m9.596s	1702m36.636s	1m33.756s
1,4,5,6,7 e 8	743MB	88,91714277	143m56.092s	71m32.836s	20m58.616s	1561m59.313s	1497m2.572s	1m39.604s
1,5,6,7 e 8	741MB	89,16204986	138m54.344s	68m13.608s	19m24.104s	1463m37.582s	1377m1.008s	1m40.204s
1,6,7 e 8	1,2GB	84,50701177	166m21.779s	86m48.940s	27m56.900s	2130m53.383s	2070m7.544s	1m44.336s
1,7 e 8	1,1GB	84,47887812	175m35.903s	84m42.104s	26m42.748s	2062m8.831s	1984m58.376s	1m48.576s
1 e 8	684KB	73,63804247	88m52.090s	33m40.800s	3m24.176s	155m12.932s	111m34.368s	1m16.328s
1,2,3,4,5,6 e 8	153MB	89,47440559	95m27.912s	44m37.024s	8m10.260s	686m7.244s	640m44.848s	1m20.452s
1,2,3,4,5 e 8	9,0MB	89,30271814	57m49.444s	37m21.344s	4m49.180s	420m37.727s	396m59.252s	1m7.784s
1,2,3,4 e 8	7,8MB	88,32633599	122m40.559s	39m29.952s	5m35.568s	388m13.910s	357m50.816s	1m9.884s
1,2,3 e 8	5,2MB	88,37430748	83m15.929s	37m17.196s	4m58.420s	458m19.187s	339m41.808s	1m25.876s
1,2 e 8	1,7MB	87,07835584	98m55.604s	35m7.360s	3m51.472s	327m49.796s	208m2.652s	1m24.420s
1,2,5 e 8	2,6MB	87,42461623	156m50.165s	35m53.292s	3m57.828s	389m26.075s	264m36.168s	1m28.868s

B.5. Experimentos Topologia Caffenet – GPU Tesla M60 / CPU

As Tabela B5 e B6 dispõem dos resultados integrais dos experimentos realizados na GPU Tesla M60 (1GPU e 2GPU, respectivamente) em paralelo com os resultados da CPU. Experimentos realizados sobre o *dataset Pornograpy2k*, observando o tempo

necessário para classificar todo o conjunto de teste. Os dados inferem que determinadas camadas influenciam diretamente no tamanho do modelo, bem como na acurácia obtida.

Tabela B5 – Experimentos impacto modificação arquitetura CNN Caffenet - GPU Tesla M60 (1GPU) / CPU

Ways	***Size***	***Accuracy***	***Time Real***	***Time User***	***Sys***	***Time Real***	***Time User***	***Sys***
Step			***Tesla M60 (1GPU)***			***CPU***		
Full	217MB	88,02732572	62m23.874s	96m1.820s	112m22.744s	1363m35.767s	1408m48.488s	113m40.568s
1,3,4,5,6,7 e 8	746MB	89,96854801	75m9.468s	105m6.752s	116m19.352s	3174m13.386s	3204m20.192s	134m6.176s
1,4,5,6,7 e 8	743MB	88,91714277	72m59.405s	105m0.756s	119m1.152s	2689m4.030s	2722m1.664s	126m33.708s
1,5,6,7 e 8	741MB	89,16204986	70m26.809s	103m48.356s	117m23.220s	2545m54.459s	2498m57.084s	129m5.092s
1,6,7 e 8	1,2GB	84,50701177	86m17.638s	113m10.516s	123m30.868s	3575m25.756s	3582m49.460s	154m58.132s
1,7 e 8	1,1GB	84,47887812	82m33.113s	110m20.900s	124m28.040s	3254m26.036s	3262m39.292s	157m28.196s
1 e 8	684KB	73,63804247	49m50.342s	88m7.904s	111m36.136s	186m44.923s	229m37.684s	110m9.032s
1,2,3,4,5,6 e 8	153MB	89,47440559	59m17.835s	96m5.804s	113m54.224s	1128m32.123s	1175m59.892s	111m32.124s
1,2,3,4,5 e 8	9,0MB	89,30271814	53m43.112s	90m33.276s	114m10.192s	716m8.978s	764m37.188s	110m4.580s
1,2,3,4 e 8	7,8MB	88,32633599	56m27.733s	92m44.408s	113m4.992s	632m5.091s	680m29.296s	109m7.740s
1,2,3 e 8	5,2MB	88,37430748	54m34.144s	90m57.268s	111m54.956s	597m46.636s	645m23.380s	108m55.148s
1,2 e 8	1,7MB	87,07835584	51m12.814s	88m48.192s	109m22.868s	356m13.737s	402m5.484s	106m37.104s
1,2,5 e 8	2,6MB	87,42461623	50m54.625s	88m9.256s	113m1.708s	553m15.187s	600m45.600s	109m57.256s
Tesla M60 (1GPU) – 8GB								

Tabela B6 – Experimentos impacto modificação arquitetura CNN Caffenet - GPU Tesla M60 (2GPU) / CPU

Ways	***Size***	***Accuracy***	***Time Real***	***Time User***	***Sys***	***Time Real***	***Time User***	***Sys***
Step			***Tesla M60 (1GPU)***			***CPU***		
Full	217MB	88,02732572	61m43.956s	96m48.456s	114m17.328s	1363m35.767s	1408m48.488s	113m40.568s
1,3,4,5,6,7 e 8	746MB	89,96854801	76m7.701s	107m3.560s	119m13.648s	3174m13.386s	3204m20.192s	134m6.176s
1,4,5,6,7 e 8	743MB	88,91714277	73m19.326s	104m51.112s	118m3.868s	2689m4.030s	2722m1.664s	126m33.708s
1,5,6,7 e 8	741MB	89,16204986	71m15.989s	103m35.876s	120m12.444s	2545m54.459s	2498m57.084s	129m5.092s
1,6,7 e 8	1,2GB	84,50701177	85m1.685s	112m42.036s	124m24.448s	3575m25.756s	3582m49.460s	154m58.132s
1,7 e 8	1,1GB	84,47887812	83m1.918s	113m15.388s	123m4.652s	3254m26.036s	3262m39.292s	157m28.196s
1 e 8	684KB	73,63804247	49m53.299s	89m45.532s	111m34.532s	186m44.923s	229m37.684s	110m9.032s
1,2,3,4,5,6 e 8	153MB	89,47440559	60m28.950s	96m16.556s	116m37.220s	1128m32.123s	1175m59.892s	111m32.124s
1,2,3,4,5 e 8	9,0MB	89,30271814	54m6.153s	92m19.348s	113m7.748s	716m8.978s	764m37.188s	110m4.580s
1,2,3,4 e 8	7,8MB	88,32633599	56m48.184s	94m2.772s	115m9.816s	632m5.091s	680m29.296s	109m7.740s
1,2,3 e 8	5,2MB	88,37430748	55m36.768s	92m54.864s	115m30.236s	597m46.636s	645m23.380s	108m55.148s
1,2 e 8	1,7MB	87,07835584	52m19.013s	90m38.408s	113m32.816s	356m13.737s	402m5.484s	106m37.104s
1,2,5 e 8	2,6MB	87,42461623	50m57.719s	89m47.820s	111m6.480s	553m15.187s	600m45.600s	109m57.256s
Tesla M60 (2GPU) – 16GB								

Apêndice C – Benchmark Caffe

Este capítulo apresenta na integra os resultados obtidos dos experimentos realizados para analisar a topologia das arquiteturas CNN modificadas do Caffenet (disposto na secção 5.5.1). Os experimentos foram realizados pela ferramenta de benchmark do framework Caffe, permitindo verificar o tempo necessário para executar cada camada individualmente e em conjunto.

C.1. Experimentos Benchmark Topologia Caffenet – GPU Titan XP / CPU

A Tabela C1 dispõem dos resultados integrais dos experimentos de benchmark realizados na GPU Titan XP em paralelo a CPU. Os experimentos foram realizados em cada uma das camadas modificadas.

Tabela C1 – Experimentos de *benchmark* Caffe – GPU Titan XP / CPU

GPU Titan XP						
Models			***Caffenet Full***		***Caffenet_1_2_3_4_5_6_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00242077ms	0.000538ms	0.00243818ms	0.000561ms
	data	***backward***	0.0024055ms	0.000715ms	0.00241693ms	0.000686ms
conv1	***conv1***	***forward***	0.104337ms	10.3076ms	0.102777ms	10.2957ms
	conv1	***backward***	0.0943096ms	10.6155ms	0.0946675ms	10.616ms
	relu1	***forward***	0.00556045ms	0.220493ms	0.00565878ms	0.220358ms
	relu1	***backward***	0.00240218ms	0.00049ms	0.00240525ms	0.000514ms
	pool1	***forward***	0.0117554ms	0.84441ms	0.0115749ms	0.842607ms
	pool1	***backward***	0.0024055ms	0.000489ms	0.00245235ms	0.000418ms
	norm1	***forward***	0.0268904ms	1.88732ms	0.026172ms	1.88794ms
	norm1	***backward***	0.145863ms	2.02764ms	0.14588ms	2.02797ms
conv2	***conv2***	***forward***	0.126ms	19.1199ms	0.127082ms	18.6371ms
	conv2	***backward***	0.103881ms	18.2764ms	0.104515ms	18.7539ms
	relu2	***forward***	0.00482317ms	0.142475ms	0.00479149ms	0.142444ms
	relu2	***backward***	0.00240115ms	0.000458ms	0.00242458ms	0.000411ms
	pool2	***forward***	0.00781376ms	0.554608ms	0.00838518ms	0.551577ms
	pool2	***backward***	0.00242301ms	0.00061ms	0.00243651ms	0.000516ms
	norm2	***forward***	0.0477696ms	2.04311ms	0.0480048ms	2.10485ms
	norm2	***backward***	0.367499ms	2.12547ms	0.36783ms	2.12356ms
conv3	***conv3***	***forward***	0.0838386ms	16.0194ms	0.0829711ms	15.9346ms
	conv3	***backward***	0.0592246ms	15.2189ms	0.059527ms	15.2366ms
	relu3	***forward***	0.00331958ms	0.050319ms	0.00359846ms	0.050356ms

	relu3	***backward***	0.00242915ms	0.000604ms	0.00243843ms	0.000669ms
conv4	***conv4***	***forward***	0.0844772ms	12.2571ms	0.0843759ms	12.2134ms
	conv4	***backward***	0.0700157ms	11.7794ms	0.0699755ms	11.7872ms
	relu4	***forward***	0.00359075ms	0.049736ms	0.00347462ms	0.049831ms
	relu4	***backward***	0.00242147ms	0.000391ms	0.00244854ms	0.000373ms
conv5	***conv5***	***forward***	0.0734683ms	8.40891ms	0.0729675ms	8.44401ms
	conv5	***backward***	0.0566219ms	8.04919ms	0.0567323ms	8.07262ms
	relu5	***forward***	0.00343878ms	0.033369ms	0.00357827ms	0.033343ms
	relu5	***backward***	0.00220781ms	0.000521ms	0.00258602ms	0.00053ms
	pool5	***forward***	0.00565994ms	0.225733ms	0.00639526ms	0.233579ms
	pool5	***backward***	0.00268038ms	0.000906ms	0.0024351ms	0.000948ms
fc6	***fc6***	***forward***	0.416825ms	47.1884ms	0.417772ms	47.3371ms
	fc6	***backward***	0.727147ms	40.3044ms	0.725223ms	40.6013ms
	relu6	***forward***	0.0035927ms	0.004967ms	0.00362368ms	0.004997ms
	relu6	***backward***	0.00245037ms	0.000464ms	0.00240618ms	0.00017ms
	drop6	***forward***	0.00729392ms	0.01967ms	0.00737453ms	0.020242ms
	drop6	***backward***	0.00238282ms	0.001047ms	0.00243914ms	0.000307ms
fc7	***fc7***	***forward***	0.184481ms	21.7995ms	x	x
	fc7	***backward***	0.329946ms	17.5647ms	x	x
	relu7	***forward***	0.00360093ms	0.004929ms	x	x
	relu7	***backward***	0.00241181ms	0.000151ms	x	x
	drop7	***forward***	0.00668861ms	0.019497ms	x	x
	drop7	***backward***	0.0024233ms	0.000355ms	x	x
fc8	***fc8***	***forward***	0.0258592ms	0.011313ms	0.0260372ms	0.011536ms
	fc8	***backward***	0.0181728ms	0.000874ms	0.018423ms	0.000868ms
prob	***prob***	***forward***	0.018486ms	0.004736ms	0.0183903ms	0.005046ms
	prob	***backward***	0.00880048ms	0.001649ms	0.00888502ms	0.00154ms
Average Forward pass			1.46949ms	141.229ms	1.25576ms	119.03ms
Average Backward pass			2.21328ms	125.983ms	1.85617ms	109.236ms
Average Forward-Backward			3.72302ms	267.245ms	3.15174ms	228.299ms
Total Time			3723.02ms	267245ms	3151.74ms	228299ms
Models			***Caffenet_1_2_3_4_5_8***		***Caffenet_1_2_3_4_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.0024329ms	0.000487ms	0.00242701ms	0.000564ms
	data	***backward***	0.0024193ms	0.000722ms	0.00242429ms	0.000635ms
conv1	***conv1***	***forward***	0.107139ms	10.3106ms	0.103309ms	10.3009ms
	conv1	***backward***	0.0961638ms	10.6727ms	0.0953473ms	10.5704ms
	relu1	***forward***	0.00567773ms	0.220445ms	0.00630458ms	0.220349ms
	relu1	***backward***	0.0024159ms	0.000466ms	0.00241318ms	0.0004ms
	pool1	***forward***	0.0119505ms	0.837713ms	0.0114944ms	0.830737ms
	pool1	***backward***	0.00242835ms	0.000471ms	0.00243667ms	0.000576ms
	norm1	***forward***	0.0273823ms	1.88974ms	0.0268957ms	1.88892ms
	norm1	***backward***	0.149651ms	2.0283ms	0.149217ms	2.03159ms
conv2	***conv2***	***forward***	0.129025ms	18.9997ms	0.12886ms	18.587ms
	conv2	***backward***	0.106351ms	18.3413ms	0.106652ms	18.3701ms

	relu2	***forward***	0.00491798ms	0.142481ms	0.00492966ms	0.142547ms
	relu2	***backward***	0.0024351ms	0.00046ms	0.00244115ms	0.000411ms
	pool2	***forward***	0.00831469ms	0.551651ms	0.00817478ms	0.551513ms
	pool2	***backward***	0.00242176ms	0.000516ms	0.00241878ms	0.000618ms
	norm2	***forward***	0.0487045ms	2.04264ms	0.0492524ms	2.04242ms
	norm2	***backward***	0.377264ms	2.1238ms	0.376063ms	2.12492ms
conv3	***conv3***	***forward***	0.0851258ms	16.0285ms	0.0846512ms	15.8962ms
	conv3	***backward***	0.0604826ms	15.2364ms	0.0601927ms	15.2351ms
	relu3	***forward***	0.00364541ms	0.050541ms	0.00352326ms	0.050479ms
	relu3	***backward***	0.00241984ms	0.000664ms	0.00242442ms	0.00039ms
conv4	***conv4***	***forward***	0.0861938ms	12.3146ms	0.0868978ms	12.166ms
	conv4	***backward***	0.0719137ms	11.7978ms	0.0702137ms	11.7468ms
	relu4	***forward***	0.00368698ms	0.049891ms	0.00357168ms	0.049719ms
	relu4	***backward***	0.0024009ms	0.000311ms	0.00245619ms	0.000291ms
conv5	***conv5***	***forward***	0.0741795ms	8.42366ms	x	x
	conv5	***backward***	0.0571089ms	8.02483ms	x	x
	relu5	***forward***	0.00337974ms	0.033275ms	x	x
	relu5	***backward***	0.00255891ms	0.000237ms	x	x
	pool5	***forward***	0.00634995ms	0.223801ms	x	x
	pool5	***backward***	0.00230093ms	0.0003ms	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.0558395ms	0.01796ms	0.351645ms	0.202058ms
	fc8	***backward***	0.0254915ms	0.001068ms	0.0279822ms	0.142751ms
prob	***prob***	***forward***	0.0186658ms	0.003801ms	0.0186711ms	0.004165ms
	prob	***backward***	0.00912698ms	0.001416ms	0.00907728ms	0.001251ms
Average Forward pass			0.842605ms	72.1491ms	1.0271ms	62.9412ms
Average Backward pass			1.12933ms	68.2392ms	1.03918ms	60.2328ms
Average Forward-Backward			2.01114ms	140.418ms	2.10699ms	123.204ms
Total Time			2011.14ms	140418ms	2106.99ms	123204ms
Models			***Caffenet_1_2_3_8***		***Caffenet_1_2_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00241754ms	0.00055ms	0.00241869ms	0.000528ms
	data	***backward***	0.00245014ms	0.000584ms	0.00243712ms	0.000747ms
conv1	***conv1***	***forward***	0.105679ms	10.2938ms	0.0996069ms	10.3138ms

	conv1	***backward***	0.0959095ms	10.5385ms	0.0928941ms	10.5772ms
	relu1	***forward***	0.00790438ms	0.220756ms	0.00586237ms	0.220587ms
	relu1	***backward***	0.0024039ms	0.000424ms	0.00243098ms	0.000446ms
	pool1	***forward***	0.013138ms	0.828883ms	0.0122212ms	0.831589ms
	pool1	***backward***	0.00242787ms	0.000454ms	0.00241152ms	0.000413ms
	norm1	***forward***	0.0279481ms	1.88911ms	0.0262256ms	1.89175ms
	norm1	***backward***	0.149919ms	2.02298ms	0.14484ms	2.0307ms
conv2	***conv2***	***forward***	0.129285ms	18.8186ms	0.124291ms	18.5708ms
	conv2	***backward***	0.107194ms	18.2917ms	0.103089ms	18.3316ms
	relu2	***forward***	0.00501008ms	0.142186ms	0.00493398ms	0.142268ms
	relu2	***backward***	0.00241456ms	0.000342ms	0.00243507ms	0.000227ms
	pool2	***forward***	0.00803667ms	0.550511ms	0.00758435ms	0.551207ms
	pool2	***backward***	0.00240477ms	0.000589ms	0.00241971ms	0.000299ms
	norm2	***forward***	0.0490298ms	2.04332ms	0.0471185ms	2.04392ms
	norm2	***backward***	0.375215ms	2.11548ms	0.333949ms	2.09859ms
conv3	***conv3***	***forward***	0.0839185ms	15.8859ms	x	x
	conv3	***backward***	0.0599189ms	15.2043ms	x	x
	relu3	***forward***	0.00369808ms	0.050191ms	x	x
	relu3	***backward***	0.00242746ms	0.000307ms	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.352378ms	0.199871ms	0.230186ms	0.138189ms
	fc8	***backward***	0.0294744ms	0.149556ms	0.0260542ms	0.100949ms
prob	***prob***	***forward***	0.0189996ms	0.003843ms	0.0181944ms	0.003588ms
	prob	***backward***	0.00935411ms	0.001308ms	0.00870915ms	0.000982ms
Average Forward pass			0.922327ms	50.9338ms	0.683842ms	34.713ms

Average Backward pass			0.952776ms	48.3318ms	0.815313ms	33.1466ms
Average Forward-Backward			1.9241ms	99.295ms	1.54158ms	67.888ms
Total Time			1924.1ms	99295ms	1541.58ms	67888ms
Models			***Caffenet_1_3_4_5_6_7_8***		***Caffenet_1_4_5_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00242074ms	0.000545ms	0.00245043ms	0.000593ms
	data	***backward***	0.00241814ms	0.000596ms	0.00237466ms	0.000666ms
conv1	***conv1***	***forward***	0.104498ms	10.5035ms	0.102495ms	10.5329ms
	conv1	***backward***	0.0969987ms	10.5211ms	0.0935898ms	10.5584ms
	relu1	***forward***	0.00530739ms	0.220438ms	0.00516243ms	0.220469ms
	relu1	***backward***	0.00241562ms	0.000336ms	0.00239411ms	0.000414ms
	pool1	***forward***	0.0115815ms	0.831324ms	0.0119877ms	0.836984ms
	pool1	***backward***	0.00241258ms	0.000558ms	0.00242918ms	0.000623ms
	norm1	***forward***	0.026292ms	1.8853ms	0.0260429ms	1.88682ms
	norm1	***backward***	0.146772ms	2.04143ms	0.147081ms	2.03659ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	0.112253ms	18.451ms	x	x
	conv3	***backward***	0.101884ms	18.5641ms	x	x
	relu3	***forward***	0.00525834ms	0.212858ms	x	x
	relu3	***backward***	0.00240602ms	0.000791ms	x	x
conv4	***conv4***	***forward***	0.206275ms	38.4065ms	0.0707276ms	9.62786ms
	conv4	***backward***	0.242969ms	38.2074ms	0.0548712ms	9.66683ms
	relu4	***forward***	0.00535389ms	0.213245ms	0.0053255ms	0.21172ms
	relu4	***backward***	0.00238198ms	0.00079ms	0.00238298ms	0.000893ms
conv5	***conv5***	***forward***	0.169852ms	26.7433ms	0.167742ms	26.764ms
	conv5	***backward***	0.140703ms	26.3185ms	0.141516ms	26.2662ms
	relu5	***forward***	0.00488608ms	0.142527ms	0.00491616ms	0.142242ms
	relu5	***backward***	0.00224106ms	0.000646ms	0.00230976ms	0.000611ms
	pool5	***forward***	0.00853603ms	0.586842ms	0.0082608ms	0.583923ms
	pool5	***backward***	0.00260467ms	0.000911ms	0.00246643ms	0.000926ms
fc6	***fc6***	***forward***	1.89345ms	216.655ms	1.89216ms	216.742ms
	fc6	***backward***	3.4838ms	187.416ms	3.48598ms	187.132ms
	relu6	***forward***	0.0036151ms	0.004901ms	0.00369338ms	0.004832ms
	relu6	***backward***	0.00251686ms	0.000453ms	0.00239901ms	0.000433ms
	drop6	***forward***	0.00730541ms	0.01895ms	0.00730109ms	0.018446ms
	drop6	***backward***	0.00233315ms	0.000827ms	0.00243917ms	0.000845ms
fc7	***fc7***	***forward***	0.183019ms	21.2157ms	0.183811ms	21.2847ms
	fc7	***backward***	0.329629ms	17.3558ms	0.330258ms	17.3598ms

	relu7	***forward***	0.00359488ms	0.00469ms	0.00361926ms	0.004724ms
	relu7	***backward***	0.0023952ms	0.000196ms	0.00244288ms	0.000175ms
	drop7	***forward***	0.00659453ms	0.018563ms	0.0066039ms	0.01828ms
	drop7	***backward***	0.00241786ms	0.000224ms	0.00238595ms	0.000226ms
fc8	***fc8***	***forward***	0.0256818ms	0.010431ms	0.0255854ms	0.010193ms
	fc8	***backward***	0.0180094ms	0.001111ms	0.0179812ms	0.000881ms
prob	***prob***	***forward***	0.127079ms	0.104788ms	0.0183948ms	0.004349ms
	prob	***backward***	0.107288ms	0.016528ms	0.00872266ms	0.001681ms
Average Forward pass			3.09816ms	336.243ms	2.70419ms	288.904ms
Average Backward pass			4.8733ms	300.46ms	4.4562ms	253.037ms
Average Forward-Backward			8.01181ms	636.737ms	7.20056ms	541.975ms
Total Time			8011.81ms	636737ms	7200.56ms	541975ms
Models			***Caffenet_1_5_6_7_8***		***Caffenet_1_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00242685ms	0.000605ms	0.00241766ms	0.000613ms
	data	***backward***	0.00240602ms	0.000611ms	0.00238182ms	0.000665ms
conv1	***conv1***	***forward***	0.10377ms	10.6038ms	0.10483ms	10.6325ms
	conv1	***backward***	0.0941248ms	10.7631ms	0.100784ms	10.5863ms
	relu1	***forward***	0.00601645ms	0.220476ms	0.00575267ms	0.220769ms
	relu1	***backward***	0.00242784ms	0.000347ms	0.00241357ms	0.000457ms
	pool1	***forward***	0.0121788ms	0.831932ms	0.0114566ms	0.831313ms
	pool1	***backward***	0.0023911ms	0.000553ms	0.002404ms	0.000634ms
	norm1	***forward***	0.0260158ms	1.88693ms	0.0263707ms	1.89105ms
	norm1	***backward***	0.148976ms	2.04461ms	0.14738ms	2.05459ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	0.0656487ms	6.67556ms	x	x
	conv5	***backward***	0.0513037ms	6.74112ms	x	x
	relu5	***forward***	0.00466765ms	0.141616ms	x	x
	relu5	***backward***	0.00236854ms	0.000573ms	x	x
	pool5	***forward***	0.0083809ms	0.552195ms	x	x

	pool5	***backward***	0.00244531ms	0.001067ms	x	x
fc6	***fc6***	***forward***	1.89395ms	217.43ms	3.19547ms	406.164ms
	fc6	***backward***	3.49277ms	189.786ms	5.6849ms	326.661ms
	relu6	***forward***	0.00361709ms	0.005001ms	0.00379235ms	0.004981ms
	relu6	***backward***	0.00245475ms	0.000473ms	0.00248115ms	0.000425ms
	drop6	***forward***	0.00724042ms	0.019233ms	0.00723222ms	0.018926ms
	drop6	***backward***	0.00236237ms	0.000964ms	0.00233472ms	0.000869ms
fc7	***fc7***	***forward***	0.18267ms	21.5084ms	0.18173ms	21.5116ms
	fc7	***backward***	0.329554ms	17.6365ms	0.329443ms	17.1781ms
	relu7	***forward***	0.00359418ms	0.0045ms	0.0036025ms	0.004703ms
	relu7	***backward***	0.00242998ms	0.000236ms	0.0023961ms	0.000248ms
	drop7	***forward***	0.0066385ms	0.018411ms	0.00664723ms	0.018337ms
	drop7	***backward***	0.00241968ms	0.000226ms	0.00242579ms	0.000209ms
fc8	***fc8***	***forward***	0.0257968ms	0.010376ms	0.0251994ms	0.010153ms
	fc8	***backward***	0.0188122ms	0.001246ms	0.0186465ms	0.001088ms
prob	***prob***	***forward***	0.127876ms	0.105612ms	0.126472ms	0.105148ms
	prob	***backward***	0.107945ms	0.017084ms	0.106614ms	0.016901ms
Average Forward pass			2.63231ms	260.023ms	3.8268ms	441.421ms
Average Backward pass			4.4107ms	227.003ms	6.52436ms	356.508ms
Average Forward-Backward			7.08451ms	487.061ms	10.3923ms	797.964ms
Total Time			7084.51ms	487061ms	10392.3ms	797964ms
Models			***Caffenet_1_7_8***		***Caffenet_1_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00242176ms	0.000632ms	0.00243197ms	0.000528ms
	data	***backward***	0.00240326ms	0.000491ms	0.00243958ms	0.000627ms
conv1	***conv1***	***forward***	0.11042ms	10.5785ms	0.102798ms	10.3073ms
	conv1	***backward***	0.100707ms	10.8199ms	0.102303ms	10.5677ms
	relu1	***forward***	0.00547437ms	0.220663ms	0.00587974ms	0.220135ms
	relu1	***backward***	0.00242381ms	0.000459ms	0.0024561ms	0.000274ms
	pool1	***forward***	0.0119357ms	0.834887ms	0.0114569ms	0.831933ms
	pool1	***backward***	0.00241843ms	0.000526ms	0.00244467ms	0.000359ms
	norm1	***forward***	0.0259289ms	1.88665ms	0.0280231ms	1.8873ms
	norm1	***backward***	0.147549ms	2.05388ms	0.145567ms	2.07459ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x

conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	3.19693ms	416.558ms	x	x
	fc7	***backward***	5.6805ms	328.073ms	x	x
	relu7	***forward***	0.00363222ms	0.005301ms	x	x
	relu7	***backward***	0.0024248ms	0.000216ms	x	x
	drop7	***forward***	0.00728144ms	0.020198ms	x	x
	drop7	***backward***	0.00244326ms	0.000404ms	x	x
fc8	***fc8***	***forward***	0.0256898ms	0.012318ms	0.389222ms	0.20088ms
	fc8	***backward***	0.0183925ms	0.001ms	0.0347862ms	0.159485ms
prob	***prob***	***forward***	0.0185069ms	0.004582ms	0.0191093ms	0.003486ms
	prob	***backward***	0.00877104ms	0.001581ms	0.0091961ms	0.001014ms
Average Forward pass			3.49608ms	430.127ms	0.62379ms	13.454ms
Average Backward pass			6.05387ms	340.957ms	0.360189ms	12.8064ms
Average Forward-Backward			9.58936ms	771.117ms	1.02545ms	26.287ms
Total Time			9589.36ms	771117ms	1025.45ms	26287ms

C.2. Experimentos Benchmark Topologia Caffenet – TK1 GPU (GK20a) / CPU

A Tabela C2 dispõem dos resultados integrais dos experimentos de benchmark realizados na NVIDIA Jetson TK1 (K20A) em paralelo a CPU. Os experimentos foram realizados em cada uma das camadas modificadas.

Tabela C2 - Experimentos de *benchmark* Caffe – Jetson TK1 / CPU

Jetson TK1 (K20A)						
Models			***Caffenet Full***		***Caffenet_1_2_3_4_5_6_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.154168ms	0.00173ms	0.12519ms	0.001687ms
	data	***backward***	0.154138ms	0.002312ms	0.130175ms	0.00218ms
conv1	***conv1***	***forward***	4.03978ms	85.2352ms	4.02179ms	84.8785ms
	conv1	***backward***	3.81428ms	67.3165ms	3.74063ms	68.0897ms
	relu1	***forward***	0.374352ms	3.48176ms	0.367681ms	2.6742ms

	relu1	***backward***	0.163554ms	0.001614ms	0.134938ms	0.001538ms
	pool1	***forward***	0.476702ms	4.32425ms	0.457859ms	4.14022ms
	pool1	***backward***	0.169832ms	0.002124ms	0.142244ms	0.002103ms
	norm1	***forward***	0.442493ms	17.7038ms	0.426767ms	17.7184ms
	norm1	***backward***	0.578526ms	17.6944ms	0.549913ms	17.9378ms
conv2	***conv2***	***forward***	10.4555ms	137.365ms	8.68335ms	137.584ms
	conv2	***backward***	5.62117ms	136.311ms	5.52362ms	137.507ms
	relu2	***forward***	0.566807ms	1.50601ms	0.69169ms	1.51563ms
	relu2	***backward***	0.231752ms	0.001712ms	0.160025ms	0.001565ms
	pool2	***forward***	1.0754ms	2.15416ms	1.15082ms	2.12686ms
	pool2	***backward***	0.229558ms	0.001999ms	0.172447ms	0.002047ms
	norm2	***forward***	0.615195ms	13.4285ms	0.955194ms	13.456ms
	norm2	***backward***	1.44378ms	13.7804ms	1.32602ms	13.7356ms
conv3	***conv3***	***forward***	3.94035ms	91.222ms	3.74472ms	90.5976ms
	conv3	***backward***	4.20837ms	92.5301ms	4.0861ms	91.7915ms
	relu3	***forward***	0.384735ms	0.54472ms	0.361123ms	0.52763ms
	relu3	***backward***	0.279825ms	0.002266ms	0.257361ms	0.00214ms
conv4	***conv4***	***forward***	5.90632ms	70.9184ms	4.50897ms	70.7962ms
	conv4	***backward***	3.49833ms	69.2034ms	3.43801ms	70.058ms
	relu4	***forward***	0.545719ms	0.524773ms	0.53039ms	0.521214ms
	relu4	***backward***	0.299611ms	0.002226ms	0.242629ms	0.002218ms
conv5	***conv5***	***forward***	5.39031ms	47.1037ms	4.86406ms	47.1156ms
	conv5	***backward***	2.71233ms	47.2066ms	2.60279ms	47.794ms
	relu5	***forward***	0.92488ms	0.350816ms	0.936756ms	0.349009ms
	relu5	***backward***	0.250297ms	0.001735ms	0.206992ms	0.001603ms
	pool5	***forward***	0.614795ms	0.779631ms	0.819486ms	0.777139ms
	pool5	***backward***	0.236441ms	0.002413ms	0.207468ms	0.002553ms
fc6	***fc6***	***forward***	18.8889ms	184.988ms	19.0698ms	185.314ms
	fc6	***backward***	28.9612ms	182.825ms	29.4005ms	181.505ms
	relu6	***forward***	0.386632ms	0.043095ms	0.277881ms	0.038118ms
	relu6	***backward***	0.0801908ms	0.001749ms	0.0658928ms	0.001115ms
	drop6	***forward***	1.39095ms	0.072919ms	1.03143ms	0.072326ms
	drop6	***backward***	0.0801135ms	0.002463ms	0.0654378ms	0.001261ms
fc7	***fc7***	***forward***	8.57139ms	81.0069ms	x	x
	fc7	***backward***	12.8401ms	133.83ms	x	x
	relu7	***forward***	0.178829ms	0.038045ms	x	x
	relu7	***backward***	0.0737959ms	0.001103ms	x	x
	drop7	***forward***	0.302394ms	0.070758ms	x	x
	drop7	***backward***	0.0721338ms	0.001274ms	x	x
fc8	***fc8***	***forward***	0.30831ms	0.056945ms	0.564394ms	0.072856ms
	fc8	***backward***	0.358704ms	0.003196ms	0.345882ms	0.003291ms
prob	***prob***	***forward***	0.633818ms	0.011405ms	0.670281ms	0.011607ms
	prob	***backward***	0.418932ms	0.00441ms	0.418397ms	0.004362ms
Average Forward pass			79.3729ms	743.015ms	67.8354ms	660.365ms
Average Backward pass			71.6672ms	760.799ms	57.1759ms	628.511ms
Average Forward-Backward			152.456ms	1504.12ms	126.262ms	1289.18ms

Total Time			152456ms	1.50412e+06ms	126262ms	1.28918e+06ms
Models			***Caffenet_1_2_3_4_5_8***		***Caffenet_1_2_3_4_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.0569927ms	0.001671ms	0.0567614ms	0.001704ms
	data	***backward***	0.0366033ms	0.002221ms	0.0294034ms	0.002194ms
conv1	***conv1***	***forward***	4.09103ms	83.7851ms	4.20429ms	83.9862ms
	conv1	***backward***	3.77035ms	68.3376ms	4.01293ms	67.6074ms
	relu1	***forward***	0.305181ms	2.66261ms	0.26262ms	2.76101ms
	relu1	***backward***	0.0432506ms	0.001526ms	0.034343ms	0.001551ms
	pool1	***forward***	0.375047ms	4.04798ms	0.353435ms	4.34564ms
	pool1	***backward***	0.0336115ms	0.002193ms	0.0307444ms	0.00223ms
	norm1	***forward***	0.233626ms	17.5243ms	0.226132ms	17.6732ms
	norm1	***backward***	0.397772ms	18.0154ms	0.368653ms	17.847ms
conv2	***conv2***	***forward***	5.44096ms	137.264ms	5.23019ms	137.187ms
	conv2	***backward***	5.54531ms	136.826ms	5.64075ms	136.417ms
	relu2	***forward***	0.330027ms	1.50311ms	0.273665ms	1.49763ms
	relu2	***backward***	0.0509442ms	0.00165ms	0.0749895ms	0.00149ms
	pool2	***forward***	0.340128ms	2.13172ms	0.479147ms	2.13131ms
	pool2	***backward***	0.0290218ms	0.001903ms	0.0192855ms	0.002003ms
	norm2	***forward***	0.301358ms	13.4943ms	0.304969ms	13.4256ms
	norm2	***backward***	1.17506ms	13.4791ms	1.20719ms	13.5512ms
conv3	***conv3***	***forward***	4.09329ms	90.8068ms	3.98125ms	90.3761ms
	conv3	***backward***	3.80227ms	90.5395ms	4.1609ms	89.2063ms
	relu3	***forward***	0.146378ms	0.525473ms	0.142327ms	0.529097ms
	relu3	***backward***	0.0231318ms	0.002262ms	0.081637ms	0.002267ms
conv4	***conv4***	***forward***	3.17119ms	70.4461ms	3.25872ms	70.5883ms
	conv4	***backward***	3.19499ms	69.2228ms	3.02033ms	68.3877ms
	relu4	***forward***	0.170474ms	0.520415ms	0.178105ms	0.525985ms
	relu4	***backward***	0.050535ms	0.002386ms	0.0870996ms	0.001564ms
conv5	***conv5***	***forward***	2.49748ms	46.9637ms	x	x
	conv5	***backward***	2.37541ms	47.1745ms	x	x
	relu5	***forward***	0.167263ms	0.346701ms	x	x
	relu5	***backward***	0.125752ms	0.001146ms	x	x
	pool5	***forward***	0.179425ms	0.773528ms	x	x
	pool5	***backward***	0.133179ms	0.001395ms	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x

	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.395543ms	0.127331ms	1.49728ms	0.903796ms
	fc8	***backward***	0.927381ms	0.093177ms	1.03562ms	0.611888ms
prob	***prob***	***forward***	0.733775ms	0.011591ms	0.531406ms	0.012985ms
	prob	***backward***	0.649743ms	0.004512ms	0.435691ms	0.004772ms
Average Forward pass			27.0363ms	473.001ms	24.7192ms	426.004ms
Average Backward pass			30.413ms	443.761ms	28.6945ms	393.691ms
Average Forward-Backward			58.2274ms	917.069ms	54.5415ms	820.003ms
Total Time			58227.4ms	917069ms	54541.5ms	820003ms
Models			***Caffenet_1_2_3_8***		***Caffenet_1_2_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.0790861ms	0.00169ms	0.063074ms	0.001726ms
	data	***backward***	0.0183372ms	0.002139ms	0.0339223ms	0.00216ms
conv1	***conv1***	***forward***	5.5546ms	83.7416ms	4.45867ms	81.9162ms
	conv1	***backward***	3.92369ms	67.7186ms	3.62362ms	67.2243ms
	relu1	***forward***	0.271147ms	2.43656ms	0.366202ms	2.48528ms
	relu1	***backward***	0.072592ms	0.001572ms	0.0448939ms	0.001555ms
	pool1	***forward***	0.3655ms	4.12514ms	0.354915ms	5.61377ms
	pool1	***backward***	0.0337532ms	0.002353ms	0.0439108ms	0.002129ms
	norm1	***forward***	0.230752ms	17.6555ms	0.338738ms	17.7841ms
	norm1	***backward***	0.429408ms	17.7372ms	0.40555ms	17.7003ms
conv2	***conv2***	***forward***	5.21623ms	137.519ms	7.4211ms	137.457ms
	conv2	***backward***	5.748ms	136.201ms	5.3697ms	136.652ms
	relu2	***forward***	0.273557ms	1.50517ms	0.466653ms	1.50013ms
	relu2	***backward***	0.140642ms	0.001662ms	0.0427613ms	0.001498ms
	pool2	***forward***	0.337028ms	2.12969ms	0.422305ms	2.12258ms
	pool2	***backward***	0.0768403ms	0.002164ms	0.0403481ms	0.001877ms
	norm2	***forward***	0.28214ms	13.424ms	0.333756ms	13.3946ms
	norm2	***backward***	1.20344ms	13.5727ms	1.23195ms	13.6625ms
conv3	***conv3***	***forward***	4.06076ms	90.7571ms	x	x
	conv3	***backward***	3.70298ms	90.1477ms	x	x
	relu3	***forward***	0.110421ms	0.524017ms	x	x
	relu3	***backward***	0.0591275ms	0.0017ms	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x

	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	1.40395ms	0.894535ms	1.05485ms	0.622629ms
	fc8	***backward***	1.09872ms	0.590425ms	1.18721ms	0.39919ms
prob	***prob***	***forward***	0.428891ms	0.013312ms	0.478446ms	0.012414ms
	prob	***backward***	0.414268ms	0.004732ms	0.640853ms	0.004202ms
Average Forward pass			23.3207ms	354.773ms	21.1559ms	262.955ms
Average Backward pass			23.2761ms	326.021ms	15.2624ms	235.681ms
Average Forward-Backward			48.2845ms	681.091ms	37.4993ms	498.93ms
Total Time			48284.5ms	681091ms	37499.3ms	498930ms
Models			***Caffenet_1_3_4_5_6_7_8***		***Caffenet_1_4_5_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	*Exception**	0.004228ms	*Exception**	0.004319ms
	data	***backward***	*Exception**	1.37715ms	*Exception**	1.6272ms
conv1	***conv1***	***forward***	*Exception**	222.762ms	*Exception**	231.121ms
	conv1	***backward***	*Exception**	110.445ms	*Exception**	119.852ms
	relu1	***forward***	*Exception**	46.6551ms	*Exception**	47.048ms
	relu1	***backward***	*Exception**	0.001818ms	*Exception**	0.00887ms
	pool1	***forward***	*Exception**	41.8434ms	*Exception**	70.2146ms
	pool1	***backward***	*Exception**	0.004985ms	*Exception**	0.024606ms
	norm1	***forward***	*Exception**	28.8965ms	*Exception**	28.0378ms
	norm1	***backward***	*Exception**	81.1054ms	*Exception**	88.6674ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	*Exception**	346.612ms	x	x
	conv3	***backward***	*Exception**	195.588ms	x	x
	relu3	***forward***	*Exception**	2.41305ms	x	x
	relu3	***backward***	*Exception**	0.014749ms	x	x
conv4	***conv4***	***forward***	*Exception**	387.99ms	*Exception**	153.816ms
	conv4	***backward***	*Exception**	430.717ms	*Exception**	161.189ms

	relu4	***forward***	*Exception**	2.26409ms	*Exception**	2.47441ms
	relu4	***backward***	*Exception**	0.019398ms	*Exception**	0.044343ms
conv5	***conv5***	***forward***	*Exception**	275.117ms	*Exception**	318.413ms
	conv5	***backward***	*Exception**	486.921ms	*Exception**	415.373ms
	relu5	***forward***	*Exception**	1.54234ms	*Exception**	1.53618ms
	relu5	***backward***	*Exception**	6.22498ms	*Exception**	0.90453ms
	pool5	***forward***	*Exception**	27.9535ms	*Exception**	10.7952ms
	pool5	***backward***	*Exception**	13.103ms	*Exception**	22.0201ms
fc6	***fc6***	***forward***	*Exception**	11030.6ms	*Exception**	8490.39ms
	fc6	***backward***	*Exception**	13976.1ms	*Exception**	9890.39ms
	relu6	***forward***	*Exception**	7.67224ms	*Exception**	3.43667ms
	relu6	***backward***	*Exception**	0.005447ms	*Exception**	0.005642ms
	drop6	***forward***	*Exception**	78.3197ms	*Exception**	85.5317ms
	drop6	***backward***	*Exception**	0.805014ms	*Exception**	0.270332ms
fc7	***fc7***	***forward***	*Exception**	1700.79ms	*Exception**	1294.79ms
	fc7	***backward***	*Exception**	1645.27ms	*Exception**	1128.69ms
	relu7	***forward***	*Exception**	0.142485ms	*Exception**	0.195758ms
	relu7	***backward***	*Exception**	0.004217ms	*Exception**	0.003238ms
	drop7	***forward***	*Exception**	1.04019ms	*Exception**	4.04256ms
	drop7	***backward***	*Exception**	2.63006ms	*Exception**	3.01992ms
fc8	***fc8***	***forward***	*Exception**	67.966ms	*Exception**	49.6987ms
	fc8	***backward***	*Exception**	0.014981ms	*Exception**	0.010794ms
prob	***prob***	***forward***	*Exception**	69.3915ms	*Exception**	46.4111ms
	prob	***backward***	*Exception**	5.93876ms	*Exception**	2.38253ms
Average Forward pass			*Exception**	14348.7ms	*Exception**	10856ms
Average Backward pass			*Exception**	16968.2ms	*Exception**	11839.3ms
Average Forward-Backward			*Exception**	31349.5ms	*Exception**	22746.1ms
Total Time			*Exception**	3.13495e+07ms	*Exception**	2.27461e+07ms

Exception - Limit Memory for load file model Caffenet_1_4_5_6_7_8 (GPU), memory required for data: 7505276*

Exception - Limit Memory for load file model Caffenet_1_3_4_5_6_7_8 (GPU), memory required for data: 9744764*

Models			***Caffenet_1_5_6_7_8***		***Caffenet_1_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	*Exception**	0.006344ms	*Exception**	0.010528ms
	data	***backward***	*Exception**	20.055ms	*Exception**	102.896ms
conv1	***conv1***	***forward***	*Exception**	319.091ms	*Exception**	866.059ms
	conv1	***backward***	*Exception**	249.031ms	*Exception**	1090.76ms
	relu1	***forward***	*Exception**	97.8305ms	*Exception**	20.8574ms
	relu1	***backward***	*Exception**	0.009997ms	*Exception**	0.051179ms
	pool1	***forward***	*Exception**	210.659ms	*Exception**	59.5876ms
	pool1	***backward***	*Exception**	0.014654ms	*Exception**	36.7673ms
	norm1	***forward***	*Exception**	30.1509ms	*Exception**	24.017ms

	norm1	***backward***	*Exception**	211.038ms	*Exception**	1079.63ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	*Exception**	454.226ms	x	x
	conv5	***backward***	*Exception**	255.796ms	x	x
	relu5	***forward***	*Exception**	1.89461ms	x	x
	relu5	***backward***	*Exception**	2.77366ms	x	x
	pool5	***forward***	*Exception**	51.7118ms	x	x
	pool5	***backward***	*Exception**	15.1544ms	x	x
fc6	***fc6***	***forward***	*Exception**	9364.16ms	*Exception**	164343ms
	fc6	***backward***	*Exception**	11290.1ms	*Exception**	142898ms
	relu6	***forward***	*Exception**	9.0164ms	*Exception**	17.8006ms
	relu6	***backward***	*Exception**	0.003737ms	*Exception**	0.006154ms
	drop6	***forward***	*Exception**	75.4358ms	*Exception**	138.351ms
	drop6	***backward***	*Exception**	0.679345ms	*Exception**	0.017044ms
fc7	***fc7***	***forward***	*Exception**	1357.63ms	*Exception**	5826.63ms
	fc7	***backward***	*Exception**	1222.71ms	*Exception**	6456.12ms
	relu7	***forward***	*Exception**	0.199357ms	*Exception**	0.089482ms
	relu7	***backward***	*Exception**	0.004044ms	*Exception**	0.004393ms
	drop7	***forward***	*Exception**	6.02998ms	*Exception**	14.5864ms
	drop7	***backward***	*Exception**	13.3948ms	*Exception**	1.86221ms
fc8	***fc8***	***forward***	*Exception**	111.062ms	*Exception**	289.92ms
	fc8	***backward***	*Exception**	4.57877ms	*Exception**	7.13497ms
prob	***prob***	***forward***	*Exception**	147.103ms	*Exception**	119.688ms
	prob	***backward***	*Exception**	23.5871ms	*Exception**	8.19625ms
Average Forward pass			*Exception**	12265.6ms	*Exception**	171724ms
Average Backward pass			*Exception**	13347.9ms	*Exception**	151690ms
Average Forward-Backward			*Exception**	25852.2ms	*Exception**	323741ms
Total Time			*Exception**	2.58522e+07ms	*Exception**	3.23741e+08ms

Exception - Limit Memory for load file model Caffenet_1_6_7_8 (GPU), memory required for data: 3599740*

Exception -Limit Memory for load file model Caffenet_1_5_6_7_8 (GPU), memory required for data: 5265788*

models			***Caffenet_1_7_8***		***Caffenet_1_8***	
ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	*Exception**	0.007616ms	0.0641003ms	0.001759ms
	data	***backward***	*Exception**	22.5398ms	0.0347888ms	0.002107ms
conv1	***conv1***	***forward***	*Exception**	906.532ms	5.56169ms	83.6922ms
	conv1	***backward***	*Exception**	1133.84ms	3.84273ms	67.4561ms
	relu1	***forward***	*Exception**	55.7949ms	0.316828ms	2.37119ms
	relu1	***backward***	*Exception**	0.011549ms	0.0504027ms	0.001587ms
	pool1	***forward***	*Exception**	65.2048ms	0.383641ms	3.79644ms
	pool1	***backward***	*Exception**	21.56ms	0.0409483ms	0.002055ms
	norm1	***forward***	*Exception**	22.5993ms	0.277474ms	17.4347ms
	norm1	***backward***	*Exception**	532.32ms	0.530776ms	17.7488ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	*Exception**	134162ms	x	x
	fc7	***backward***	*Exception**	112732ms	x	x

	relu7	***forward***	*Exception**	6.24042ms	x	x
	relu7	***backward***	*Exception**	0.004078ms	x	x
	drop7	***forward***	*Exception**	56.9529ms	x	x
	drop7	***backward***	*Exception**	0.071696ms	x	x
fc8	***fc8***	***forward***	*Exception**	40.6856ms	1.57457ms	1.0426ms
	fc8	***backward***	*Exception**	0.530906ms	1.53284ms	0.637463ms
prob	***prob***	***forward***	*Exception**	8.48217ms	0.415812ms	0.013294ms
	prob	***backward***	*Exception**	0.045161ms	0.558387ms	0.004506ms
Average Forward pass			*Exception**	135325ms	12.2947ms	108.376ms
Average Backward pass			*Exception**	114445ms	9.95112ms	85.8708ms
Average Forward-Backward			*Exception**	249984ms	23.4655ms	194.539ms
Total Time			*Exception**	2.49984e+08ms	23465.5ms	194539ms

Exception - Limit Memory for load file model Caffenet_1_7_8 (GPU), memory required for data: 3550588*

C.3. Experimentos Benchmark Topologia Caffenet – GeForce GTX 750 Ti / CPU

A Tabela C3 dispõem dos resultados integrais dos experimentos de benchmark realizados na GeForce GTX 750 Ti em paralelo a CPU. Os experimentos foram realizados em cada uma das camadas modificadas.

Tabela C3 - Experimentos de *benchmark* Caffe - GeForce GTX 750 Ti / CPU

GeForce GTX 750 Ti						
Models			***Caffenet Full***		***Caffenet_1_2_3_4_5_6_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00096832ms	0.000593ms	0.000969664ms	0.000515ms
	data	***backward***	0.000960864ms	0.000658ms	0.000962624ms	0.000596ms
conv1	***conv1***	***forward***	0.351711ms	10.501ms	0.35383ms	10.3822ms
	conv1	***backward***	0.459274ms	10.6279ms	0.45964ms	10.5084ms
	relu1	***forward***	0.0208812ms	0.221982ms	0.0205522ms	0.220152ms
	relu1	***backward***	0.000961696ms	0.000329ms	0.000960192ms	0.000359ms
	pool1	***forward***	0.0466013ms	0.982114ms	0.0464895ms	0.967712ms
	pool1	***backward***	0.00096288ms	0.000443ms	0.000960896ms	0.000426ms
	norm1	***forward***	0.0496807ms	1.92263ms	0.0497482ms	1.89476ms
	norm1	***backward***	0.195037ms	2.0637ms	0.195197ms	2.03246ms
conv2	***conv2***	***forward***	0.557588ms	18.8458ms	0.558248ms	18.5897ms
	conv2	***backward***	0.577957ms	18.5427ms	0.57771ms	18.2749ms
	relu2	***forward***	0.0152141ms	0.143803ms	0.0152358ms	0.142279ms
	relu2	***backward***	0.000958272ms	0.000372ms	0.000959712ms	0.000325ms
	pool2	***forward***	0.0315034ms	0.658683ms	0.0315604ms	0.646714ms
	pool2	***backward***	0.000961824ms	0.000547ms	0.00096256ms	0.00061ms
	norm2	***forward***	0.0762553ms	2.11247ms	0.0764351ms	2.06813ms
	norm2	***backward***	0.492983ms	2.15764ms	0.492251ms	2.13536ms

conv3	***conv3***	***forward***	0.419557ms	16.1158ms	0.42062ms	15.8682ms
	conv3	***backward***	0.327458ms	15.4698ms	0.325789ms	15.2094ms
	relu3	***forward***	0.00898598ms	0.051078ms	0.00908451ms	0.05081ms
	relu3	***backward***	0.000959808ms	0.000642ms	0.000963904ms	0.000715ms
conv4	***conv4***	***forward***	0.394675ms	12.388ms	0.394729ms	12.1584ms
	conv4	***backward***	0.347741ms	12.0276ms	0.347058ms	11.763ms
	relu4	***forward***	0.00890922ms	0.050324ms	0.00892886ms	0.049852ms
	relu4	***backward***	0.000960384ms	0.000511ms	0.000961248ms	0.000487ms
conv5	***conv5***	***forward***	0.29395ms	8.53379ms	0.294194ms	8.36109ms
	conv5	***backward***	0.231394ms	8.26341ms	0.23158ms	8.03719ms
	relu5	***forward***	0.00777709ms	0.034159ms	0.00766733ms	0.033393ms
	relu5	***backward***	0.000957056ms	0.000545ms	0.000958656ms	0.000519ms
	pool5	***forward***	0.0136495ms	0.255068ms	0.0137196ms	0.252749ms
	pool5	***backward***	0.000978048ms	0.000863ms	0.000983456ms	0.000883ms
fc6	***fc6***	***forward***	2.22991ms	47.9894ms	2.22996ms	47.0265ms
	fc6	***backward***	4.46038ms	44.07ms	4.45277ms	43.9676ms
	relu6	***forward***	0.00553776ms	0.004957ms	0.00545104ms	0.004813ms
	relu6	***backward***	0.000961216ms	0.000686ms	0.000965088ms	0.000256ms
	drop6	***forward***	0.0201866ms	0.020931ms	0.0200802ms	0.019042ms
	drop6	***backward***	0.000957408ms	0.000859ms	0.00096272ms	0.000315ms
fc7	***fc7***	***forward***	0.944382ms	22.0537ms	x	x
	fc7	***backward***	2.14655ms	19.5864ms	x	x
	relu7	***forward***	0.00544861ms	0.0048ms	x	x
	relu7	***backward***	0.000959936ms	0.000281ms	x	x
	drop7	***forward***	0.0186389ms	0.020393ms	x	x
	drop7	***backward***	0.000950592ms	0.000246ms	x	x
fc8	***fc8***	***forward***	0.0562072ms	0.011216ms	0.056185ms	0.01088ms
	fc8	***backward***	0.0282364ms	0.000923ms	0.0275844ms	0.000828ms
prob	***prob***	***forward***	0.029198ms	0.004915ms	0.0292972ms	0.004604ms
	prob	***backward***	0.0175681ms	0.001736ms	0.0170763ms	0.001821ms
Average Forward pass			5.71419ms	142.938ms	4.73711ms	118.762ms
Average Backward pass			9.40699ms	132.828ms	7.23281ms	111.945ms
Average Forward-Backward			15.1493ms	275.8ms	11.996ms	230.739ms
Total Time			15149.3ms	275800ms	11996ms	230739ms
Models			***Caffenet_1_2_3_4_5_8***		***Caffenet_1_2_3_4_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.000964992ms	0.000487ms	0.000960928ms	0.000763ms
	data	***backward***	0.000964832ms	0.000635ms	0.000962752ms	0.000636ms
conv1	***conv1***	***forward***	0.350103ms	10.3521ms	0.350048ms	10.3687ms
	conv1	***backward***	0.461817ms	10.5863ms	0.461434ms	10.5168ms
	relu1	***forward***	0.0207718ms	0.220062ms	0.0206319ms	0.220066ms
	relu1	***backward***	0.000964768ms	0.000365ms	0.000964736ms	0.000374ms
	pool1	***forward***	0.0467067ms	0.966654ms	0.0464985ms	0.968927ms
	pool1	***backward***	0.00096512ms	0.000435ms	0.000962208ms	0.000404ms
	norm1	***forward***	0.050053ms	1.89931ms	0.0497383ms	1.89615ms

	norm1	***backward***	0.195977ms	2.03762ms	0.195413ms	2.03363ms
conv2	***conv2***	***forward***	0.559213ms	18.6022ms	0.557754ms	18.5919ms
	conv2	***backward***	0.579157ms	18.3778ms	0.577365ms	18.2582ms
	relu2	***forward***	0.01538ms	0.142459ms	0.0153285ms	0.142176ms
	relu2	***backward***	0.000963552ms	0.000404ms	0.00096288ms	0.000344ms
	pool2	***forward***	0.0316284ms	0.648378ms	0.0315384ms	0.647745ms
	pool2	***backward***	0.000960032ms	0.000569ms	0.000967936ms	0.000596ms
	norm2	***forward***	0.0765726ms	2.07172ms	0.0763404ms	2.06744ms
	norm2	***backward***	0.49339ms	2.14614ms	0.493013ms	2.14026ms
conv3	***conv3***	***forward***	0.422404ms	15.9515ms	0.42025ms	15.8712ms
	conv3	***backward***	0.32668ms	15.2566ms	0.329485ms	15.2349ms
	relu3	***forward***	0.00905562ms	0.051032ms	0.00898416ms	0.050304ms
	relu3	***backward***	0.000963424ms	0.000545ms	0.000964704ms	0.000486ms
conv4	***conv4***	***forward***	0.39429ms	12.2215ms	0.394395ms	12.1742ms
	conv4	***backward***	0.348126ms	11.78ms	0.348111ms	11.7333ms
	relu4	***forward***	0.0090152ms	0.049858ms	0.00897437ms	0.04981ms
	relu4	***backward***	0.000960128ms	0.000402ms	0.0009648ms	0.000292ms
conv5	***conv5***	***forward***	0.29613ms	8.38732ms	x	x
	conv5	***backward***	0.229675ms	8.02663ms	x	x
	relu5	***forward***	0.00791043ms	0.033476ms	x	x
	relu5	***backward***	0.000959104ms	0.000257ms	x	x
	pool5	***forward***	0.0139538ms	0.249598ms	x	x
	pool5	***backward***	0.000963776ms	0.000249ms	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.115095ms	0.017632ms	0.747584ms	0.197138ms
	fc8	***backward***	0.0354897ms	0.000886ms	0.105728ms	0.142867ms
prob	***prob***	***forward***	0.028943ms	0.004601ms	0.0290028ms	0.004879ms
	prob	***backward***	0.0172121ms	0.001738ms	0.0172634ms	0.001776ms
Average Forward pass			2.52637ms	71.877ms	2.82549ms	63.2578ms
Average Backward pass			2.77659ms	68.2237ms	2.60634ms	60.0706ms
Average Forward-Backward			5.32941ms	140.134ms	5.45725ms	123.361ms
Total Time			5329.41ms	140134ms	5457.25ms	123361ms
Models			***Caffenet_1_2_3_8***		***Caffenet_1_2_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***

data	***data***	***forward***	0.00100742ms	0.000592ms	0.000999968ms	0.000455ms
	data	***backward***	0.00100685ms	0.000626ms	0.000968736ms	0.000688ms
conv1	***conv1***	***forward***	0.356003ms	10.3304ms	0.353029ms	10.3962ms
	conv1	***backward***	0.461729ms	10.5715ms	0.460482ms	10.6677ms
	relu1	***forward***	0.0207994ms	0.220061ms	0.020858ms	0.220064ms
	relu1	***backward***	0.000978144ms	0.000432ms	0.000962784ms	0.000407ms
	pool1	***forward***	0.0470417ms	0.968037ms	0.0471349ms	0.96808ms
	pool1	***backward***	0.00099792ms	0.000457ms	0.000961792ms	0.000497ms
	norm1	***forward***	0.0500473ms	1.90098ms	0.0502033ms	1.89518ms
	norm1	***backward***	0.195845ms	2.03408ms	0.197ms	2.03276ms
conv2	***conv2***	***forward***	0.559936ms	18.5918ms	0.562912ms	18.8718ms
	conv2	***backward***	0.580835ms	18.4034ms	0.580417ms	18.5203ms
	relu2	***forward***	0.0154356ms	0.142148ms	0.0154673ms	0.142185ms
	relu2	***backward***	0.000994528ms	0.000339ms	0.000965248ms	0.00026ms
	pool2	***forward***	0.0319637ms	0.646731ms	0.0322218ms	0.647721ms
	pool2	***backward***	0.000984032ms	0.000631ms	0.000984416ms	0.000361ms
	norm2	***forward***	0.076714ms	2.08612ms	0.0769244ms	2.06815ms
	norm2	***backward***	0.493777ms	2.13242ms	0.448828ms	2.11593ms
conv3	***conv3***	***forward***	0.420897ms	15.8872ms	x	x
	conv3	***backward***	0.318887ms	15.214ms	x	x
	relu3	***forward***	0.0092201ms	0.050354ms	x	x
	relu3	***backward***	0.000990848ms	0.00037ms	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.747207ms	0.191204ms	0.504371ms	0.130871ms
	fc8	***backward***	0.104902ms	0.143417ms	0.0773612ms	0.094317ms

prob	***prob***	***forward***	0.0296545ms	0.004661ms	0.029214ms	0.003482ms
	prob	***backward***	0.0179147ms	0.001488ms	0.0174382ms	0.000907ms
Average Forward pass			2.42546ms	51.0261ms	1.74402ms	35.3485ms
Average Backward pass			2.24046ms	48.5078ms	1.83591ms	33.4379ms
Average Forward-Backward			4.69258ms	99.566ms	3.60536ms	68.815ms
Total Time			4692.58ms	99566ms	3605.36ms	68815ms
Models			***Caffenet_1_3_4_5_6_7_8***		***Caffenet_1_4_5_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.000968544ms	0.000488ms	0.000971776ms	0.000589ms
	data	***backward***	0.000975392ms	0.000808ms	0.000968128ms	0.000665ms
conv1	***conv1***	***forward***	0.348006ms	10.488ms	0.34779ms	10.373ms
	conv1	***backward***	0.456748ms	10.5685ms	0.45906ms	10.52ms
	relu1	***forward***	0.0205718ms	0.22092ms	0.020581ms	0.220178ms
	relu1	***backward***	0.000959136ms	0.000326ms	0.000968512ms	0.000321ms
	pool1	***forward***	0.0464518ms	0.976346ms	0.0463655ms	0.969216ms
	pool1	***backward***	0.000960608ms	0.000512ms	0.000962656ms	0.000502ms
	norm1	***forward***	0.0502616ms	1.90629ms	0.0496455ms	1.8969ms
	norm1	***backward***	0.196818ms	2.04411ms	0.194282ms	2.03334ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	0.469095ms	18.5556ms	x	x
	conv3	***backward***	0.505307ms	18.4855ms	x	x
	relu3	***forward***	0.0198707ms	0.213439ms	x	x
	relu3	***backward***	0.000964224ms	0.000987ms	x	x
conv4	***conv4***	***forward***	1.24767ms	38.8135ms	0.43719ms	9.62472ms
	conv4	***backward***	1.2148ms	38.4ms	0.461183ms	9.61237ms
	relu4	***forward***	0.020024ms	0.214228ms	0.0199937ms	0.211771ms
	relu4	***backward***	0.000958624ms	0.000937ms	0.000962592ms	0.000811ms
conv5	***conv5***	***forward***	0.773591ms	27.0702ms	0.77132ms	26.8188ms
	conv5	***backward***	0.776288ms	26.4827ms	0.779586ms	26.3907ms
	relu5	***forward***	0.0153356ms	0.143228ms	0.0152691ms	0.143733ms
	relu5	***backward***	0.000957504ms	0.000607ms	0.00096304ms	0.000636ms
	pool5	***forward***	0.0315612ms	0.676227ms	0.0314956ms	0.674926ms
	pool5	***backward***	0.00096672ms	0.00097ms	0.000969184ms	0.000981ms
fc6	***fc6***	***forward***	10.3601ms	219.828ms	10.3461ms	218.73ms
	fc6	***backward***	20.8536ms	190.847ms	20.8513ms	195.373ms
	relu6	***forward***	0.00590288ms	0.004941ms	0.00592861ms	0.004915ms
	relu6	***backward***	0.000956896ms	0.000617ms	0.0009664ms	0.000489ms
	drop6	***forward***	0.0207051ms	0.019143ms	0.0204514ms	0.019089ms

	drop6	***backward***	0.0009624ms	0.00085ms	0.00096256ms	0.000846ms
fc7	***fc7***	***forward***	0.927531ms	21.6457ms	0.931741ms	21.5074ms
	fc7	***backward***	2.06407ms	19.4382ms	2.05722ms	19.4532ms
	relu7	***forward***	0.00567462ms	0.004585ms	0.00544438ms	0.004636ms
	relu7	***backward***	0.000964224ms	0.00028ms	0.000962976ms	0.000193ms
	drop7	***forward***	0.0188649ms	0.018555ms	0.0190274ms	0.018636ms
	drop7	***backward***	0.0009608ms	0.000254ms	0.000963616ms	0.000337ms
fc8	***fc8***	***forward***	0.0559168ms	0.010284ms	0.0564999ms	0.010508ms
	fc8	***backward***	0.0280266ms	0.001252ms	0.0280311ms	0.000888ms
prob	***prob***	***forward***	0.428793ms	0.107658ms	0.0297925ms	0.00488ms
	prob	***backward***	0.239658ms	0.017146ms	0.016915ms	0.001761ms
Average Forward pass			14.9671ms	340.928ms	13.2353ms	291.243ms
Average Backward pass			26.4409ms	306.302ms	24.9435ms	263.399ms
Average Forward-Backward			41.4355ms	647.264ms	38.2061ms	554.675ms
Total Time			41435.5ms	647264ms	38206.1ms	554675ms
Models			***Caffenet_1_5_6_7_8***		***Caffenet_1_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.000979488ms	0.000541ms	*Exception**	0.000547ms
	data	***backward***	0.000963552ms	0.000617ms	*Exception**	0.000554ms
conv1	***conv1***	***forward***	0.355484ms	10.3678ms	*Exception**	10.3369ms
	conv1	***backward***	0.458444ms	10.6305ms	*Exception**	10.6377ms
	relu1	***forward***	0.0205275ms	0.220036ms	*Exception**	0.220507ms
	relu1	***backward***	0.00096128ms	0.000218ms	*Exception**	0.000437ms
	pool1	***forward***	0.0468032ms	0.970094ms	*Exception**	0.968709ms
	pool1	***backward***	0.000962784ms	0.000359ms	*Exception**	0.00053ms
	norm1	***forward***	0.0495302ms	1.89643ms	*Exception**	1.90054ms
	norm1	***backward***	0.194362ms	2.03487ms	*Exception**	2.048ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	0.279577ms	6.67066ms	x	x
	conv5	***backward***	0.331219ms	6.69856ms	x	x

	relu5	***forward***	0.015226ms	0.141619ms	x	x
	relu5	***backward***	0.00096096ms	0.000479ms	x	x
	pool5	***forward***	0.0316331ms	0.648953ms	x	x
	pool5	***backward***	0.000966592ms	0.000969ms	x	x
fc6	***fc6***	***forward***	10.3557ms	221.805ms	*Exception**	380.831ms
	fc6	***backward***	20.8518ms	192.241ms	*Exception**	310.931ms
	relu6	***forward***	0.00562493ms	0.004798ms	*Exception**	0.004907ms
	relu6	***backward***	0.000961536ms	0.000511ms	*Exception**	0.000518ms
	drop6	***forward***	0.0198146ms	0.018962ms	*Exception**	0.018742ms
	drop6	***backward***	0.000958688ms	0.00102ms	*Exception**	0.000917ms
fc7	***fc7***	***forward***	0.951247ms	21.6557ms	*Exception**	21.8535ms
	fc7	***backward***	2.05536ms	19.6094ms	*Exception**	20.7763ms
	relu7	***forward***	0.00576826ms	0.004471ms	*Exception**	0.004782ms
	relu7	***backward***	0.00096624ms	0.000221ms	*Exception**	0.000202ms
	drop7	***forward***	0.0189655ms	0.018601ms	*Exception**	0.018382ms
	drop7	***backward***	0.000963392ms	0.00029ms	*Exception**	0.000294ms
fc8	***fc8***	***forward***	0.0553941ms	0.009972ms	*Exception**	0.010179ms
	fc8	***backward***	0.0280366ms	0.001109ms	*Exception**	0.000943ms
prob	***prob***	***forward***	0.428732ms	0.106038ms	*Exception**	0.004561ms
	prob	***backward***	0.23922ms	0.016691ms	*Exception**	0.00174ms
Average Forward pass			12.7207ms	264.548ms	*Exception**	416.179ms
Average Backward pass			24.244ms	231.244ms	*Exception**	344.404ms
Average Forward-Backward			36.9917ms	495.824ms	*Exception**	760.616ms
Total Time			36991.7ms	495824ms	*Exception**	760616ms

Exception - Limit Memory for load file model Caffenet_1_6_7_8 (GPU)*

models			***Caffenet_1_7_8***		***Caffenet_1_8***	
ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	*Exception**	0.000516ms	0.000999424ms	0.000393ms
	data	***backward***	*Exception**	0.00052ms	0.000966976ms	0.000425ms
conv1	***conv1***	***forward***	*Exception**	10.3423ms	0.356193ms	10.4809ms
	conv1	***backward***	*Exception**	10.7541ms	0.462468ms	10.508ms
	relu1	***forward***	*Exception**	0.219936ms	0.021324ms	0.220004ms
	relu1	***backward***	*Exception**	0.000435ms	0.00096448ms	0.000334ms
	pool1	***forward***	*Exception**	0.967194ms	0.0476696ms	0.966392ms
	pool1	***backward***	*Exception**	0.000503ms	0.000975936ms	0.000355ms
	norm1	***forward***	*Exception**	1.89405ms	0.0506286ms	1.89832ms
	norm1	***backward***	*Exception**	2.05621ms	0.196874ms	2.01882ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x

conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	*Exception**	378.322ms	x	x
	fc7	***backward***	*Exception**	310.22ms	x	x
	relu7	***forward***	*Exception**	0.004989ms	x	x
	relu7	***backward***	*Exception**	0.000219ms	x	x
	drop7	***forward***	*Exception**	0.019263ms	x	x
	drop7	***backward***	*Exception**	0.000348ms	x	x
fc8	***fc8***	***forward***	*Exception**	0.01089ms	0.808496ms	0.207563ms
	fc8	***backward***	*Exception**	0.000911ms	0.114374ms	0.150364ms
prob	***prob***	***forward***	*Exception**	0.005109ms	0.0312329ms	0.003565ms
	prob	***backward***	*Exception**	0.00174ms	0.0179366ms	0.001118ms
Average Forward pass			*Exception**	391.791ms	1.35165ms	13.7794ms
Average Backward pass			*Exception**	323.039ms	0.826061ms	12.6812ms
Average Forward-Backward			*Exception**	714.864ms	2.20309ms	26.488ms
Total Time			*Exception**	714864ms	2203.09ms	26488ms
Exception - Limit Memory for load file model Caffenet_1_7_8 (GPU)*						

C.4. Experimentos Benchmark Topologia Caffenet – GPU Tesla M60 / CPU

A Tabela C4 dispõem dos resultados integrais dos experimentos de benchmark realizados na GPU Tesla M60 em paralelo a CPU. Os experimentos foram realizados em cada uma das camadas modificadas.

Tabela C4 - Experimentos de *benchmark* Caffe - Tesla M60

Tesla M60		
Models	***Caffenet Full***	***Caffenet_1_2_3_4_5_6_8***

Ways			*GPU*	*CPU*	*GPU*	*CPU*
data	***data***	***forward***	0.00266173ms	0.001381ms	0.00252442ms	0.001688ms
	data	***backward***	0.00276896ms	0.000823ms	0.00252154ms	0.000823ms
conv1	***conv1***	***forward***	0.218263ms	19.5082ms	0.214044ms	19.4972ms
	conv1	***backward***	0.203519ms	20.4236ms	0.204512ms	20.5607ms
	relu1	***forward***	0.0151973ms	0.419482ms	0.0159212ms	0.419272ms
	relu1	***backward***	0.00271574ms	0.000582ms	0.0025161ms	0.000476ms
	pool1	***forward***	0.0301883ms	1.59787ms	0.0302418ms	1.60158ms
	pool1	***backward***	0.00273834ms	0.000668ms	0.00251734ms	0.000769ms
	norm1	***forward***	0.0491682ms	3.09927ms	0.0495739ms	3.093ms
	norm1	***backward***	0.237537ms	3.25005ms	0.237375ms	3.38762ms
conv2	***conv2***	***forward***	0.269931ms	34.7279ms	0.269701ms	35.8913ms
	conv2	***backward***	0.360026ms	34.8158ms	0.361829ms	34.5821ms
	relu2	***forward***	0.0124634ms	0.269409ms	0.0132671ms	0.269371ms
	relu2	***backward***	0.00270595ms	0.000525ms	0.00251066ms	0.000623ms
	pool2	***forward***	0.0240794ms	1.07153ms	0.0245582ms	1.07168ms
	pool2	***backward***	0.00273286ms	0.000718ms	0.00250864ms	0.00096ms
	norm2	***forward***	0.0852281ms	3.45241ms	0.0857786ms	3.45283ms
	norm2	***backward***	0.590562ms	3.5132ms	0.590016ms	3.6106ms
conv3	***conv3***	***forward***	0.233464ms	30.1987ms	0.234499ms	30.4978ms
	conv3	***backward***	0.16646ms	28.9949ms	0.167476ms	29.009ms
	relu3	***forward***	0.0102811ms	0.0945ms	0.0109745ms	0.094462ms
	relu3	***backward***	0.0027703ms	0.000738ms	0.00253216ms	0.000704ms
conv4	***conv4***	***forward***	0.205285ms	23.1443ms	0.207618ms	23.3848ms
	conv4	***backward***	0.184614ms	22.3961ms	0.18601ms	22.4003ms
	relu4	***forward***	0.0101006ms	0.094352ms	0.0108058ms	0.094281ms
	relu4	***backward***	0.00274461ms	0.000781ms	0.00251222ms	0.000705ms
conv5	***conv5***	***forward***	0.160196ms	15.9199ms	0.161569ms	16.0421ms
	conv5	***backward***	0.145785ms	15.3703ms	0.146403ms	15.4093ms
	relu5	***forward***	0.0097232ms	0.063372ms	0.0104019ms	0.063347ms
	relu5	***backward***	0.0027216ms	0.001156ms	0.00251914ms	0.000909ms
	pool5	***forward***	0.0157028ms	0.464484ms	0.0164832ms	0.454504ms
	pool5	***backward***	0.00274445ms	0.00156ms	0.00252163ms	0.001505ms
fc6	***fc6***	***forward***	1.10576ms	78.9848ms	1.1043ms	81.7151ms
	fc6	***backward***	2.31805ms	64.3001ms	2.31514ms	65.0325ms
	relu6	***forward***	0.00906096ms	0.009072ms	0.0107804ms	0.009576ms
	relu6	***backward***	0.00273328ms	0.000883ms	0.00251795ms	0.000417ms
	drop6	***forward***	0.0206731ms	0.038209ms	0.0212517ms	0.038525ms
	drop6	***backward***	0.00273904ms	0.001462ms	0.00252858ms	0.000508ms
fc7	***fc7***	***forward***	0.487175ms	34.0493ms	x	x
	fc7	***backward***	1.0145ms	29.6104ms	x	x
	relu7	***forward***	0.00876013ms	0.008308ms	x	x
	relu7	***backward***	0.00269834ms	0.000462ms	x	x
	drop7	***forward***	0.0179595ms	0.035506ms	x	x
	drop7	***backward***	0.00272419ms	0.00049ms	x	x
fc8	***fc8***	***forward***	0.0477795ms	0.017858ms	0.0486753ms	0.021553ms

	fc8	***backward***	0.033314ms	0.00134ms	0.0346921ms	0.001628ms
prob	***prob***	***forward***	0.0428149ms	0.008655ms	0.0447424ms	0.009555ms
	prob	***backward***	0.0249429ms	0.002654ms	0.0260954ms	0.002969ms
Average Forward pass			3.33615ms	247.296ms	2.78846ms	217.737ms
Average Backward pass			5.55424ms	222.705ms	4.49441ms	194.017ms
Average Forward-Backward			8.95589ms	470.078ms	7.34438ms	411.825ms
Total Time			8955.89ms	470078ms	7344.38ms	411825ms
Models			***Caffenet_1_2_3_4_5_8***		***Caffenet_1_2_3_4_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00254726ms	0.000839ms	0.00266189ms	0.000749ms
	data	***backward***	0.00255578ms	0.00061ms	0.00275475ms	0.000587ms
conv1	***conv1***	***forward***	0.221378ms	19.501ms	0.222886ms	19.5068ms
	conv1	***backward***	0.212627ms	19.8859ms	0.212195ms	20.0574ms
	relu1	***forward***	0.0156238ms	0.418338ms	0.0167539ms	0.418408ms
	relu1	***backward***	0.00255245ms	0.000421ms	0.0027495ms	0.000412ms
	pool1	***forward***	0.0311302ms	1.58932ms	0.0326772ms	1.58875ms
	pool1	***backward***	0.00255427ms	0.000567ms	0.00275818ms	0.000508ms
	norm1	***forward***	0.051022ms	3.09359ms	0.0520756ms	3.09397ms
	norm1	***backward***	0.248154ms	3.19839ms	0.248321ms	3.21165ms
conv2	***conv2***	***forward***	0.279716ms	34.6668ms	0.280687ms	34.7892ms
	conv2	***backward***	0.374853ms	34.6507ms	0.379901ms	34.7954ms
	relu2	***forward***	0.0130668ms	0.269354ms	0.0138229ms	0.269216ms
	relu2	***backward***	0.00255757ms	0.000501ms	0.00274403ms	0.000446ms
	pool2	***forward***	0.0251456ms	1.07431ms	0.0265076ms	1.06571ms
	pool2	***backward***	0.00259638ms	0.000672ms	0.00275843ms	0.000646ms
	norm2	***forward***	0.0894931ms	3.45752ms	0.0901407ms	3.46847ms
	norm2	***backward***	0.616921ms	3.46631ms	0.615621ms	3.47769ms
conv3	***conv3***	***forward***	0.245563ms	30.0963ms	0.247539ms	30.1323ms
	conv3	***backward***	0.174241ms	28.8918ms	0.174286ms	28.9614ms
	relu3	***forward***	0.0111116ms	0.094369ms	0.0114937ms	0.094638ms
	relu3	***backward***	0.00254291ms	0.000489ms	0.00276675ms	0.000489ms
conv4	***conv4***	***forward***	0.214977ms	23.0896ms	0.217879ms	23.0908ms
	conv4	***backward***	0.192548ms	22.3005ms	0.19151ms	22.3313ms
	relu4	***forward***	0.0105153ms	0.094345ms	0.0113601ms	0.094429ms
	relu4	***backward***	0.00257213ms	0.000579ms	0.00281878ms	0.000404ms
conv5	***conv5***	***forward***	0.167557ms	15.8738ms	x	x
	conv5	***backward***	0.149725ms	15.2521ms	x	x
	relu5	***forward***	0.0103511ms	0.063175ms	x	x
	relu5	***backward***	0.00254851ms	0.000416ms	x	x
	pool5	***forward***	0.017057ms	0.443191ms	x	x
	pool5	***backward***	0.00255456ms	0.000414ms	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x

	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.0953324ms	0.028552ms	0.570014ms	0.366927ms
	fc8	***backward***	0.0417211ms	0.001162ms	0.0650499ms	0.283146ms
prob	***prob***	***forward***	0.0442491ms	0.005037ms	0.0468847ms	0.005505ms
	prob	***backward***	0.025345ms	0.001322ms	0.0267697ms	0.00132ms
Average Forward pass			1.71877ms	133.87ms	2.00245ms	117.995ms
Average Backward pass			2.23498ms	127.661ms	2.08722ms	113.13ms
Average Forward-Backward			4.01317ms	261.58ms	4.15662ms	231.171ms
Total Time			4013.17ms	261580ms	4156.62ms	231171ms
Models			***Caffenet_1_2_3_8***		***Caffenet_1_2_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00256912ms	0.000749ms	0.00256426ms	0.00068ms
	data	***backward***	0.00257491ms	0.000743ms	0.00258579ms	0.00048ms
conv1	***conv1***	***forward***	0.221269ms	19.5189ms	0.226261ms	19.5245ms
	conv1	***backward***	0.214972ms	20.4351ms	0.220321ms	19.9597ms
	relu1	***forward***	0.015555ms	0.418914ms	0.0164341ms	0.418709ms
	relu1	***backward***	0.00258045ms	0.000501ms	0.00255731ms	0.000423ms
	pool1	***forward***	0.031664ms	1.5873ms	0.0322541ms	1.57621ms
	pool1	***backward***	0.00257578ms	0.000705ms	0.00256102ms	0.000523ms
	norm1	***forward***	0.0516255ms	3.10223ms	0.0526532ms	3.39759ms
	norm1	***backward***	0.249733ms	3.22541ms	0.256748ms	3.24323ms
conv2	***conv2***	***forward***	0.281819ms	34.8004ms	0.288746ms	34.766ms
	conv2	***backward***	0.37821ms	34.7809ms	0.384358ms	34.871ms
	relu2	***forward***	0.0131137ms	0.271289ms	0.0136785ms	0.269685ms
	relu2	***backward***	0.00257322ms	0.000623ms	0.0025593ms	0.000422ms
	pool2	***forward***	0.0252664ms	1.07128ms	0.0261546ms	1.0633ms
	pool2	***backward***	0.00257946ms	0.000622ms	0.00256144ms	0.000488ms
	norm2	***forward***	0.0902937ms	3.46267ms	0.0927196ms	3.62857ms
	norm2	***backward***	0.625067ms	3.47555ms	0.595322ms	3.46091ms
conv3	***conv3***	***forward***	0.246845ms	30.2922ms	x	x
	conv3	***backward***	0.172277ms	29.023ms	x	x
	relu3	***forward***	0.0107469ms	0.094812ms	x	x
	relu3	***backward***	0.00257686ms	0.000512ms	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x

	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	x	x	x	x
	fc7	***backward***	x	x	x	x
	relu7	***forward***	x	x	x	x
	relu7	***backward***	x	x	x	x
	drop7	***forward***	x	x	x	x
	drop7	***backward***	x	x	x	x
fc8	***fc8***	***forward***	0.575425ms	0.363956ms	0.39772ms	0.2374ms
	fc8	***backward***	0.0624809ms	0.287447ms	0.0540396ms	0.185609ms
prob	***prob***	***forward***	0.0441054ms	0.00468ms	0.0452353ms	0.003647ms
	prob	***backward***	0.0283824ms	0.001242ms	0.0259664ms	0.001093ms
Average Forward pass			1.73831ms	94.9975ms	1.30256ms	64.8941ms
Average Backward pass			1.87443ms	91.2396ms	1.65987ms	61.7302ms
Average Forward-Backward			3.6715ms	186.282ms	3.02134ms	126.657ms
Total Time			3671.5ms	186282ms	3021.34ms	126657ms
Models			***Caffenet_1_3_4_5_6_7_8***		***Caffenet_1_4_5_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00263827ms	0.001571ms	0.00251914ms	0.001449ms
	data	***backward***	0.0027151ms	0.001079ms	0.0025232ms	0.00095ms
conv1	***conv1***	***forward***	0.208502ms	19.5596ms	0.209231ms	19.5243ms
	conv1	***backward***	0.19842ms	20.3378ms	0.197846ms	20.3476ms
	relu1	***forward***	0.0146004ms	0.419947ms	0.0155231ms	0.419738ms
	relu1	***backward***	0.00271645ms	0.000488ms	0.00251702ms	0.000527ms
	pool1	***forward***	0.0298523ms	1.60187ms	0.0297118ms	1.59411ms
	pool1	***backward***	0.00270605ms	0.00075ms	0.00253242ms	0.000752ms
	norm1	***forward***	0.047702ms	3.09432ms	0.0479915ms	3.10519ms
	norm1	***backward***	0.23351ms	3.28092ms	0.231787ms	3.27269ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x

conv3	***conv3***	***forward***	0.240876ms	35.553ms	x	x
	conv3	***backward***	0.239883ms	34.9205ms	x	x
	relu3	***forward***	0.014114ms	0.40518ms	x	x
	relu3	***backward***	0.00271594ms	0.000971ms	x	x
conv4	***conv4***	***forward***	0.497792ms	73.2819ms	0.222529ms	18.2515ms
	conv4	***backward***	0.59142ms	72.0379ms	0.163978ms	18.3769ms
	relu4	***forward***	0.0139574ms	0.403438ms	0.0146077ms	0.405606ms
	relu4	***backward***	0.00271606ms	0.000951ms	0.00251955ms	0.001074ms
conv5	***conv5***	***forward***	0.40293ms	50.8065ms	0.407251ms	49.8291ms
	conv5	***backward***	0.471933ms	49.9179ms	0.472799ms	49.9721ms
	relu5	***forward***	0.0122877ms	0.269465ms	0.0129198ms	0.269463ms
	relu5	***backward***	0.00270976ms	0.0009ms	0.00252826ms	0.000826ms
	pool5	***forward***	0.0236454ms	1.13747ms	0.0240406ms	1.13536ms
	pool5	***backward***	0.00271904ms	0.001741ms	0.00252736ms	0.0019ms
fc6	***fc6***	***forward***	5.08636ms	389.939ms	5.08867ms	383.132ms
	fc6	***backward***	11.1015ms	301.492ms	11.1033ms	299.42ms
	relu6	***forward***	0.00995315ms	0.009376ms	0.0114516ms	0.009698ms
	relu6	***backward***	0.00271574ms	0.000808ms	0.0025344ms	0.000927ms
	drop6	***forward***	0.0203036ms	0.037971ms	0.0210329ms	0.038702ms
	drop6	***backward***	0.00272406ms	0.001545ms	0.00252864ms	0.001815ms
fc7	***fc7***	***forward***	0.490016ms	36.2567ms	0.487953ms	35.7807ms
	fc7	***backward***	1.00877ms	31.0324ms	1.01171ms	30.7688ms
	relu7	***forward***	0.00896426ms	0.009203ms	0.00963658ms	0.009357ms
	relu7	***backward***	0.00269731ms	0.000497ms	0.00252003ms	0.000439ms
	drop7	***forward***	0.0173814ms	0.036372ms	0.018217ms	0.03669ms
	drop7	***backward***	0.00271994ms	0.000557ms	0.00252547ms	0.000422ms
fc8	***fc8***	***forward***	0.0467928ms	0.018934ms	0.04724ms	0.018708ms
	fc8	***backward***	0.0328939ms	0.001477ms	0.0340516ms	0.001462ms
prob	***prob***	***forward***	0.0430542ms	0.009732ms	0.0445348ms	0.009461ms
	prob	***backward***	0.02483ms	0.00248ms	0.0259677ms	0.002824ms
Average Forward pass			7.43413ms	612.869ms	6.88782ms	513.588ms
Average Backward pass			14.1321ms	513.049ms	13.4364ms	422.186ms
Average Forward-Backward			21.6316ms	1126ms	20.3861ms	935.856ms
Total Time			21631.6ms	1.126e+06ms	20386.1ms	935856ms
Models			***Caffenet_1_5_6_7_8***		***Caffenet_1_6_7_8***	
Ways			***GPU***	***CPU***	***GPU***	***CPU***
data	***data***	***forward***	0.00250944ms	0.001563ms	0.00266582ms	0.001667ms
	data	***backward***	0.0025144ms	0.000843ms	0.00276154ms	0.000804ms
conv1	***conv1***	***forward***	0.211009ms	19.5532ms	0.207098ms	19.5087ms
	conv1	***backward***	0.19766ms	20.7176ms	0.197942ms	20.6325ms
	relu1	***forward***	0.0150712ms	0.419929ms	0.0149473ms	0.419866ms
	relu1	***backward***	0.00250883ms	0.000478ms	0.0027329ms	0.000927ms
	pool1	***forward***	0.0294074ms	1.59708ms	0.0289961ms	1.60188ms
	pool1	***backward***	0.0024953ms	0.000642ms	0.00275766ms	0.001203ms
	norm1	***forward***	0.0476947ms	3.10022ms	0.0471875ms	3.10076ms

	norm1	***backward***	0.232115ms	3.27123ms	0.23133ms	3.28522ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	0.13881ms	12.7156ms	x	x
	conv5	***backward***	0.142729ms	12.7044ms	x	x
	relu5	***forward***	0.0125316ms	0.269308ms	x	x
	relu5	***backward***	0.00250048ms	0.001207ms	x	x
	pool5	***forward***	0.0238241ms	1.0883ms	x	x
	pool5	***backward***	0.00249347ms	0.00141ms	x	x
fc6	***fc6***	***forward***	5.09387ms	383.823ms	8.2611ms	610.85ms
	fc6	***backward***	11.1101ms	299.864ms	18.1622ms	501.519ms
	relu6	***forward***	0.0113415ms	0.009727ms	0.0101272ms	0.010225ms
	relu6	***backward***	0.00249779ms	0.00112ms	0.00276176ms	0.000973ms
	drop6	***forward***	0.0207616ms	0.03881ms	0.0200806ms	0.037627ms
	drop6	***backward***	0.00248787ms	0.001542ms	0.00275942ms	0.001209ms
fc7	***fc7***	***forward***	0.486207ms	35.823ms	0.484121ms	35.7239ms
	fc7	***backward***	1.01185ms	30.6915ms	1.01044ms	30.4474ms
	relu7	***forward***	0.00924758ms	0.009357ms	0.00883402ms	0.009575ms
	relu7	***backward***	0.00249805ms	0.000463ms	0.0027375ms	0.000456ms
	drop7	***forward***	0.0178947ms	0.03701ms	0.0174767ms	0.036793ms
	drop7	***backward***	0.00250755ms	0.000407ms	0.00275149ms	0.000442ms
fc8	***fc8***	***forward***	0.0469431ms	0.018678ms	0.0460174ms	0.021521ms
	fc8	***backward***	0.033545ms	0.001527ms	0.0324338ms	0.001779ms
prob	***prob***	***forward***	0.0439292ms	0.009221ms	0.0417012ms	0.010086ms
	prob	***backward***	0.0251332ms	0.002207ms	0.0239468ms	0.003323ms
Average Forward pass			6.36495ms	458.527ms	9.32391ms	671.344ms
Average Backward pass			12.9245ms	367.273ms	19.8078ms	555.906ms
Average Forward-Backward			19.3515ms	825.884ms	29.1958ms	1227.33ms
Total Time			19351.5ms	825884ms	29195.8ms	1.22733e+06ms
models			***Caffenet_1_7_8***		***Caffenet_1_8***	
ways			***GPU***	***CPU***	***GPU***	***CPU***

data	***data***	***forward***	0.00267238ms	0.001624ms	0.00281018ms	0.000654ms
	data	***backward***	0.0027697ms	0.000734ms	0.00288714ms	0.000497ms
conv1	***conv1***	***forward***	0.206861ms	19.4994ms	0.243534ms	19.562ms
	conv1	***backward***	0.198272ms	20.7846ms	0.238819ms	20.0174ms
	relu1	***forward***	0.0144229ms	0.419563ms	0.0165315ms	0.418305ms
	relu1	***backward***	0.00276122ms	0.000905ms	0.00287338ms	0.000502ms
	pool1	***forward***	0.0297915ms	1.59047ms	0.033578ms	1.57338ms
	pool1	***backward***	0.00276832ms	0.001131ms	0.00287763ms	0.00046ms
	norm1	***forward***	0.0469524ms	3.09614ms	0.0564801ms	3.09594ms
	norm1	***backward***	0.231499ms	3.2867ms	0.276787ms	3.17842ms
conv2	***conv2***	***forward***	x	x	x	x
	conv2	***backward***	x	x	x	x
	relu2	***forward***	x	x	x	x
	relu2	***backward***	x	x	x	x
	pool2	***forward***	x	x	x	x
	pool2	***backward***	x	x	x	x
	norm2	***forward***	x	x	x	x
	norm2	***backward***	x	x	x	x
conv3	***conv3***	***forward***	x	x	x	x
	conv3	***backward***	x	x	x	x
	relu3	***forward***	x	x	x	x
	relu3	***backward***	x	x	x	x
conv4	***conv4***	***forward***	x	x	x	x
	conv4	***backward***	x	x	x	x
	relu4	***forward***	x	x	x	x
	relu4	***backward***	x	x	x	x
conv5	***conv5***	***forward***	x	x	x	x
	conv5	***backward***	x	x	x	x
	relu5	***forward***	x	x	x	x
	relu5	***backward***	x	x	x	x
	pool5	***forward***	x	x	x	x
	pool5	***backward***	x	x	x	x
fc6	***fc6***	***forward***	x	x	x	x
	fc6	***backward***	x	x	x	x
	relu6	***forward***	x	x	x	x
	relu6	***backward***	x	x	x	x
	drop6	***forward***	x	x	x	x
	drop6	***backward***	x	x	x	x
fc7	***fc7***	***forward***	8.25547ms	612.919ms	x	x
	fc7	***backward***	18.1491ms	504.06ms	x	x
	relu7	***forward***	0.0100053ms	0.009696ms	x	x
	relu7	***backward***	0.0027617ms	0.000457ms	x	x
	drop7	***forward***	0.019514ms	0.037607ms	x	x
	drop7	***backward***	0.00277162ms	0.000486ms	x	x
fc8	***fc8***	***forward***	0.0478013ms	0.021717ms	0.685699ms	0.360428ms
	fc8	***backward***	0.0326971ms	0.001576ms	0.0727278ms	0.293127ms

prob	***prob***	***forward***	0.0417822ms	0.008707ms	0.0457388ms	0.003284ms
	prob	***backward***	0.0239281ms	0.002858ms	0.0263699ms	0.001019ms
Average Forward pass			8.77965ms	637.612ms	1.15908ms	25.0181ms
Average Backward pass			18.7504ms	528.146ms	0.696735ms	23.4951ms
Average Forward-Backward			27.5941ms	1165.84ms	1.91688ms	48.541ms
Total Time			27594.1ms	1.16584e+06ms	1916.88ms	48541ms

C.5. Baseline Benchmark Topologia Caffenet

A Tabela C5 dispõem dos resultados integrais dos experimentos de benchmark realizados na GPU Tesla M60 em paralelo a CPU. Os experimentos foram realizados em cada uma das camadas modificadas.

Tabela C5 – *Baseline benchmark* topologia *Caffenet*

Titan XP (ms)				
Models	***GPU***	***baseline***	***CPU***	***baseline***
Caffenet Full	3723,02	0	267245	0
Caffenet_1_2_3_4_5_6_8	3151,74	571,28	228299	38946
Caffenet_1_2_3_4_5_8	2011,14	1711,88	140418	126827
Caffenet_1_2_3_4_8	2106,99	1616,03	123204	144041
Caffenet_1_2_3_8	1924,1	1798,92	99295	167950
Caffenet_1_2_8	1541,58	2181,44	67888	199357
Caffenet_1_3_4_5_6_7_8	8011,81	-4288,79	636737	-369492
Caffenet_1_4_5_6_7_8	7200,56	-3477,54	541975	-274730
Caffenet_1_5_6_7_8	7084,51	-3361,49	487061	-219816
Caffenet_1_6_7_8	10392,3	-6669,28	797964	-530719
Caffenet_1_7_8	9589,36	-5866,34	771117	-503872
Caffenet_1_8	1025,45	2697,57	26287	240958
Jetson TK1 (ms)				
Models	***GPU***	***baseline***	***CPU***	***baseline***
Caffenet Full	152456	0	1,50412E+11	0E+00
Caffenet_1_2_3_4_5_6_8	126262	26194	1,28918E+11	2E+10
Caffenet_1_2_3_4_5_8	58227,4	94228,6	9,17069E+05	2E+11
Caffenet_1_2_3_4_8	54541,5	97914,5	8,20003E+05	2E+11
Caffenet_1_2_3_8	48284,5	104171,5	6,81091E+05	2E+11
Caffenet_1_2_8	37499,3	114956,7	4,98930E+05	2E+11
Caffenet_1_3_4_5_6_7_8	*Exception**	-	3,13495E+12	-3E+12
Caffenet_1_4_5_6_7_8	*Exception**	-	2,27461E+12	-2E+12
Caffenet_1_5_6_7_8	*Exception**	-	2,58522E+12	-2E+12
Caffenet_1_6_7_8	*Exception**	-	3,23741E+13	-3E+13
Caffenet_1_7_8	*Exception**	-	2,49984E+13	-2E+13
Caffenet_1_8	23465,5	128990,5	1,94539E+05	2E+11
GeForce GTX 750 Ti (ms)				

Models	***GPU***	***baseline***	***CPU***	***baseline***
Caffenet Full	15149,3	137306,7	275800	1,50412E+11
Caffenet_1_2_3_4_5_6_8	11996	140460	230739	1,50412E+11
Caffenet_1_2_3_4_5_8	5329,41	147126,59	140134	1,50412E+11
Caffenet_1_2_3_4_8	5457,25	146998,75	123361	1,50412E+11
Caffenet_1_2_3_8	4692,58	147763,42	99566	1,50412E+11
Caffenet_1_2_8	3605,36	148850,64	68815	1,50412E+11
Caffenet_1_3_4_5_6_7_8	41435,5	111020,5	647264	1,50411E+11
Caffenet_1_4_5_6_7_8	38206,1	114249,9	554675	1,50411E+11
Caffenet_1_5_6_7_8	36991,7	115464,3	495824	1,50412E+11
Caffenet_1_6_7_8	*Exception**	-	760616	1,50411E+11
Caffenet_1_7_8	*Exception**	-	714864	1,50411E+11
Caffenet_1_8	2203,09	150252,91	26488	1,50412E+11
Tesla M60 (ms)				
Models	***GPU***	***baseline***	***CPU***	***baseline***
Caffenet Full	8955,89	-5233	470078	-202833
Caffenet_1_2_3_4_5_6_8	7344,38	-3621,36	411825	-144580
Caffenet_1_2_3_4_5_8	4013,17	-290,15	261580	5665
Caffenet_1_2_3_4_8	4156,62	-433,60	231171	36074
Caffenet_1_2_3_8	3671,50	51,52	186282	80963
Caffenet_1_2_8	3021,34	701,68	126657	140588
Caffenet_1_3_4_5_6_7_8	21631,60	-17908,58	1,13E+09	-1125732755
Caffenet_1_4_5_6_7_8	20386,10	-16663,08	935856	-668611
Caffenet_1_5_6_7_8	19351,50	-15628,48	825884	-558639
Caffenet_1_6_7_8	29195,80	-25472,78	1,23E+11	-1,22733E+11
Caffenet_1_7_8	27594,10	-23871,08	1,17E+11	-1,16584E+11
Caffenet_1_8	1916,88	1806,14	48541	218704