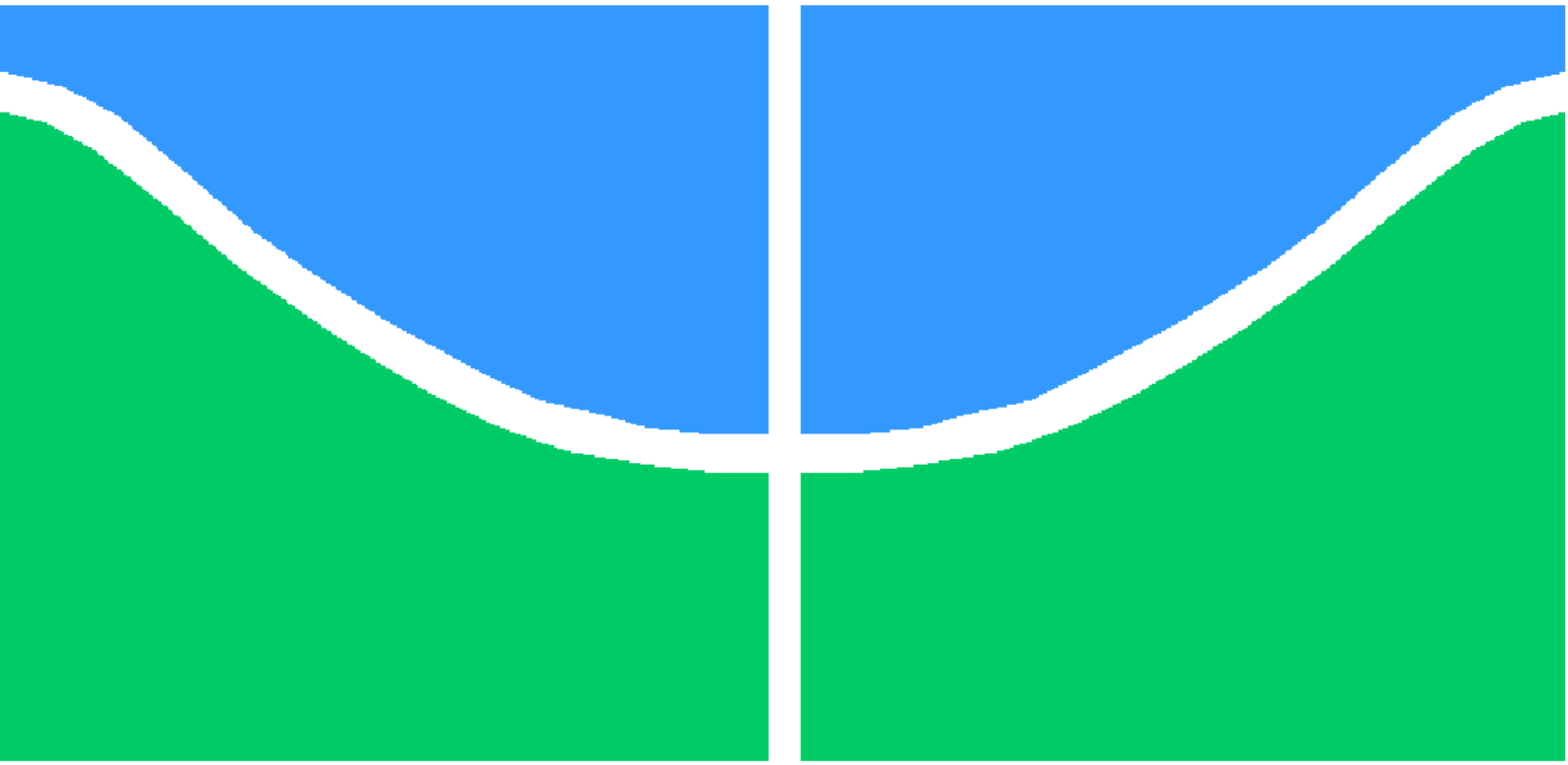


Universidade de Brasília - UnB

Faculdade de Tecnologia UnB - FT

Engenharia Elétrica

Deep Learning aplicado à Classificação de Vídeos

Autores: Gustavo Henrique Takahashi de Aquino Carvalho e Pedro Caiafa Marques

Orientador: Professor Doutor Alexandre Ricardo Soares Romariz

Brasília, DF

2018

Gustavo Henrique Takahashi de Aquino Carvalho e Pedro Caiafa Marques

Deep Learning aplicado à Classificação de Vídeos

Monografia submetida ao curso de graduação em Engenharia Elétrica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Elétrica.

Universidade de Brasília - UnB

Faculdade de Tecnologia UnB - FT

Orientador: Professor Doutor Alexandre Ricardo Soares Romariz

Brasília, DF

2018

Gustavo Henrique Takahashi de Aquino Carvalho e Pedro Caiafa Marques
Deep Learning aplicado à Classificação de Vídeos/ Gustavo Henrique Takahashi de Aquino Carvalho e Pedro Caiafa Marques. – Brasília, DF, 2018-
72 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Alexandre Ricardo Soares Romariz

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade de Tecnologia UnB - FT , 2018.

1. Classificação de ações. 2. *Deep Learning*. I. Professor Doutor Alexandre Ricardo Soares Romariz. II. Universidade de Brasília. III. Faculdade de Tecnologia UnB . IV. Deep Learning aplicado à Classificação de Vídeos

CDU 02:141:005.6

Gustavo Henrique Takahashi de Aquino Carvalho e Pedro Caiafa Marques

Deep Learning aplicado à Classificação de Vídeos

Monografia submetida ao curso de graduação em Engenharia Elétrica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Elétrica.

Trabalho aprovado. Brasília, DF, 10 de dezembro de 2018:

Professor Doutor Alexandre Ricardo Soares Romariz
Orientador

Professor Doutor Eduardo Peixoto Fernandes da Silva
ENE - UnB

Professor Doutor João Luiz Azevedo de Carvalho
ENE - UnB

Brasília, DF
2018

Resumo

A identificação automática de ações em vídeos é uma área extensamente explorada em inteligência artificial, impulsionada pela enorme quantidade de dados e tempo necessário para realização manual desta tarefa. Neste cenário, realiza-se estudo a respeito de redes capazes de classificar vídeos de acordo com as ações tomadas utilizando o *data set* UCF101, composto por 13320 vídeos divididos em 101 categorias. Comparam-se metodologias ingênuas, que realizam classificação da ação por apenas uma imagem estática, com técnicas que incorporam as informações temporais por meio de diversas arquiteturas, aferindo impactos da relação temporal na classificação. Com este fim, são utilizadas redes convolucionais, implementadas em python com auxílio da biblioteca Keras utilizando o *Tensorflow* como *backend*. É explorada a possibilidade de realizar o presente estudo com limitações de recursos expressivas quando comparadas às tecnologias disponíveis em trabalhos estado da arte. Ao final, comprova-se a viabilidade deste tipo de pesquisa com a capacidade de obter redes que, ainda que não alcancem $98,0\%$ de acurácia do melhor resultado já reportado, apresentem resultados bastante expressivos, atingindo a marca de $92,6\%$ de acurácia.

Palavras-chaves: classificação autônoma. *deep learning*. vídeo.

Abstract

Automatically identifying actions in videos is extensively explored by the usage of artificial intelligence given the vast video data there is and the time it takes to manually perform this task. With this in perspective, we set out to study networks capable of classifying videos in respect to the actions taken within them, to that end the data set UCF101, comprised of 133320 video clips divided among 101 classes, is explored. We compare trivial methodologies, which try to classify the video using only a single static image, with networks that incorporate the temporal information, hoping to asses the influence of temporal relation in the classification process. To do so convolutional networks are implemented in python with the support of the Keras library running on top of Tensorflow. We also address the viability of studying these problems with severe resource limitation compared to the technology available in most state of the art researches. We manage to confirm the possibility of researches with these limitations by training a neural network that, even though it falls short from state of the art accuracy of 98,0%, has an expressive accuracy of 92,6%.

Key-words: autonomous classification. deep learning. video.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CNN	Convolutional Neural Network (Redes Convolucionais)
CPU	Central Process Unit (Unidade Central de Processamento)
GB	Gigabyte
GPU	Graphics Processing Unit (Unidade de Processamento Gráfico)
LSTM	Long Short-Term Memory (Módulo recorrente utilizado)
MSE	Mean Squared Error (Erro Médio Quadrático)
ReLU	Rectified Linear Unit (Unidade Linear Retificada)

Sumário

1 Introdução

Técnicas de aprendizado de máquina encontram-se presentes em inúmeras tecnologias comerciais modernas, impulsionadas por consideráveis avanços recentes no desempenho desses algoritmos com o uso de métodos de aprendizado profundo, ou *deep learning*. Impactando diariamente a experiência de usuários nas mais diversas áreas, esses sistemas já são realidade sob aplicações como reconhecimento de imagens, transcrição de texto, sugestão de produtos em sites de *e-commerce* e filmes, vídeos e músicas em plataformas de *streaming*, direcionamento de conteúdo em redes sociais e resultados de busca relevantes em plataformas de pesquisa (LECUN; BENGIO; HINTON, 2015).

Abordagens convencionais de aprendizado de máquina necessitam de processamento prévio dos dados, exigindo auxílio de especialistas da área a fim de extrair informações relevantes dos dados brutos (LECUN; BENGIO; HINTON, 2015). Somente após este árduo processo aplica-se então metodologias de classificação automática. Por outro lado, métodos de *deep learning* abstraem etapa de extração manual de informações, permitindo a utilização de dados em sua forma bruta e o aprendizado automático de representações necessárias a solução do problema em questão (LECUN; BENGIO; HINTON, 2015).

Com avanços significativos nos últimos anos, o uso de aprendizado profundo permitiu melhora expressiva em múltiplas áreas, definindo o novo estado da arte em reconhecimento de voz, reconhecimento visual de objetos, descobrimento de drogas e genômica (LECUN; BENGIO; HINTON, 2015). Sob contexto deste trabalho, destaca-se a ruptura obtida em reconhecimento de imagens no trabalho de Krizhevsky, Sutskever e Hinton (2012), reduzindo a taxa de erros de 26,2% para 15,3% ao utilizar abordagem por *deep learning* e instaurando uma leva de novas otimizações.

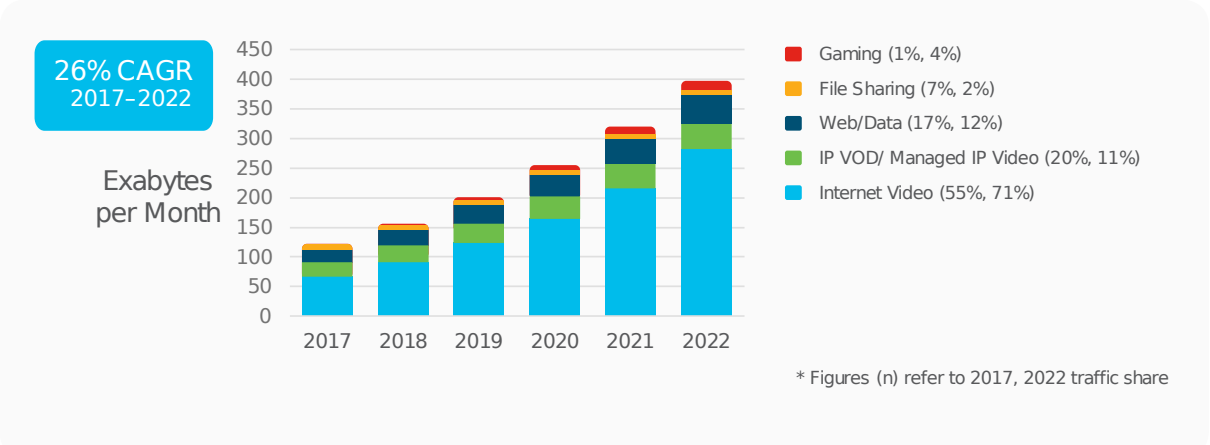


Figura 1 – Tráfego global na Internet por categoria de aplicação. Por (CISCO, 2018).

Assim, é natural que esforços bem sucedidos no campo da classificação de imagens sejam estendidos a sequências de imagens, isto é, vídeos. Ademais, relatórios (CISCO, 2018) enfatizam a importância dos estudos em vídeos, que devem compor 82% do tráfego global de Internet até 2022, acima dos 75% em 2017 (Figura 1). Destaca-se também o tráfego de vigilância por vídeo, que deve aumentar sete vezes entre 2017 e 2022, consumindo 3% de todo o tráfego de vídeo da Internet até 2022 (CISCO, 2018).

Com 90% do volume de dados digitais criados entre 2014 e 2016, até o último ano, e apenas 1 por cento destes dados analisados (HENKE; LIBARIKIAN; WISEMAN, 2016), destaca-se a necessidade do desenvolvimento de ferramentas de análise automática para fins de vigilância, busca em serviços de *streaming*, recomendação e direcionamento de conteúdo, robótica e demais aplicações, já que a inspeção humana desta massa de dados é inviável.

Neste contexto, realizou-se estudo a respeito de implementações propostas para classificação de vídeos com o intuito de construir uma rede neural capaz de gerar predições adequadas. Em busca de uma implementação viável para esta tarefa, sob contexto de pesquisa independente, foram exploradas diversas arquiteturas — redes convolucionais 2D e 3D bem como redes recorrentes —, baseando-se em estudos publicados previamente.

1.1 Definição do Problema

É abordada neste trabalho a classificação automatizada de vídeos por meio de redes neurais artificiais. Para exploração dos conceitos e aferição dos resultados, utiliza-se o *data set* UCF101, disponibilizado pela *University of Central Florida* (SOOMRO; ZAMIR; SHAH, 2012), composto de clipes de vídeo retirados do *YouTube* com o objetivo de reconhecimento de ações humanas. O estudo é realizado com recursos próprios dos autores a fim de determinar a viabilidade de pesquisas independentes na área. Este trabalho foi desenvolvido na linguagem de programação Python utilizando a biblioteca Keras (CHOLLET et al., 2015) com Tensorflow (ABADI et al., 2015) como *back-end*.

1.2 Objetivos e Resultados do Projeto

Tem-se como objetivo desenvolver uma rede que apresente desempenho compatível com o estado da arte no aspecto de classificação de vídeos de acordo com as ações tomadas. Dadas as limitações de *hardware*, há considerável distância entre as possibilidades de implementação neste trabalho e o que se tem como estado da arte. Ao longo do projeto são apresentados alguns aspectos em que esta limitação influenciou o desenvolvimento, mas mostra-se que esta realização é viável e que obter resultados expressivos é possível.

De fato, obteve-se 92,6% de acurácia no *data set* UCF101 em comparação aos 98

pontos percentuais do melhor resultado já reportado, utilizando, no entanto, consideravelmente menos recursos. Ademais, verifica-se a importância do *transfer learning* para obter resultados expressivos, bem como as diversas performances de cada arquitetura.

1.3 Estrutura do Texto

O capítulo 2 apresenta explicações das bases gerais e fundamentações para o aprendizado de máquina, destacando também técnicas de *deep learning*. O capítulo 3 apresenta abordagens de aprendizado profundo em classificação de vídeos, destacando trabalhos que abordam este problema e arquiteturas exploradas. O Capítulo 4 traz o desenvolvimento do trabalho em cima dos *data sets*, com aspectos práticos, experimentos realizados, resultados obtidos e discussão dos mesmos. O capítulo 5 conclui o trabalho com uma síntese do desenvolvimento realizado, comentários finais e propostas de trabalhos futuros.

2 Aprendizado de Máquina

2.1 Rede Neural Artificial

2.1.1 Inspiração Biológica

O desenvolvimento de redes neurais artificiais é originalmente relacionado a tentativa de modelar simplificadamente o funcionamento de neurônios humanos (KARPATHY, 2018). Em alto nível de abstração dentro do que é conhecido a respeito de neurônios biológicos, tem-se em estruturas conhecidas como dendritos a coleta de sinais de neurônios vizinhos, propagando a informação processada por meio de seu axônio. Tal axônio, por sua vez, conecta-se a subsequentes neurônios em uma estrutura sináptica, excitando ou inibindo a atividade dos neurônios conectados.

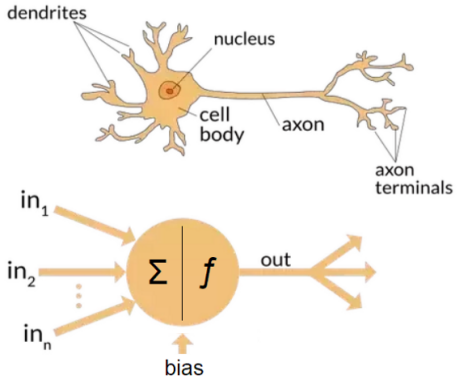


Figura 2 – Analogia entre modelo biológico e matemático. Por (SHARMA, 2017).

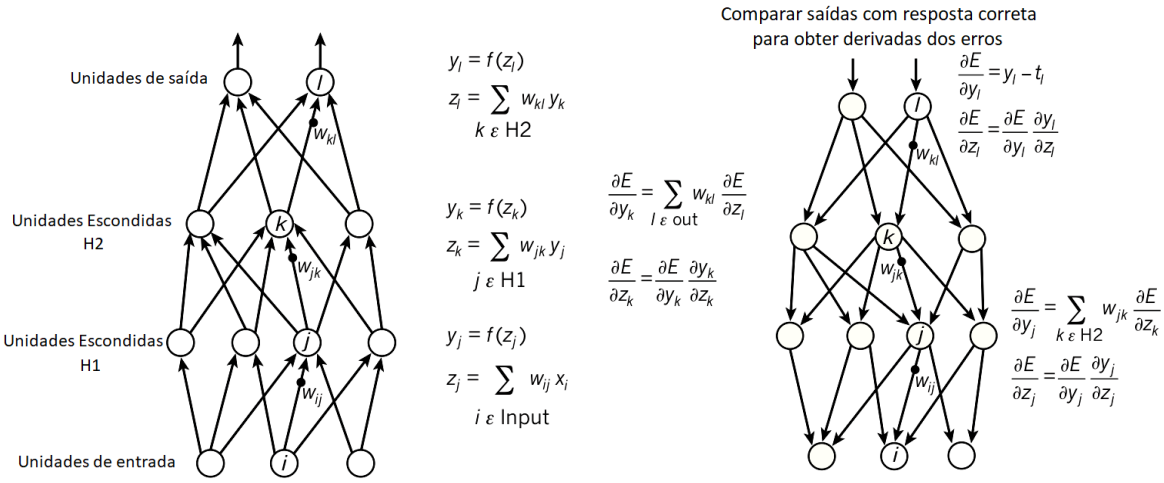
O grau de influência sináptica de um neurônio com seu subsequente varia ao longo do tempo e é compreendido como parte do processo de aprendizagem. O fenômeno resultante dessas interações consiste na emissão de um sinal elétrico ao longo do axônio de um neurônio quando excitações suficientes são recebidas por meio dos dendritos. Este processo de troca de informação foi originalmente modelado por McCulloch e Pitts (1990):

$$y = f(z), \text{tal que } z = \sum_{i}^{N} w_i x_i + b, \tag{2.1}$$

em que um sinal de entrada x interage com os pesos sinápticos w nos dendritos, compondo o sinal de saída y do axônio após a interação com uma função de ativação f. Observa-se na Figura 2 a comparação entre o neurônio biológico e seu respectivo modelo matemático simplificado.

2.1.2 Perceptron Multicamadas

Dada a proposição do modelo neural descrito pela Equação 2.1, e sob trabalhos subsequentes com a proposta do Perceptron (ROSENBLATT, 1958) e a viabilidade da utilização de múltiplas camadas com técnicas de otimização consolidadas no trabalho de Rumelhart, Hinton e Williams (1986), desenvolveu-se o que atualmente é conhecido como o perceptron multicamadas (*multilayer perceptron*, MLP).



(a) Propagação direta (b) Retro-propagação

Figura 3 – Perceptron multicamadas. Adaptado de (LECUN; BENGIO; HINTON, 2015).

Nesta abordagem, permite-se a realização de mapeamentos não-lineares com a composição de múltiplas camadas de neurônios conectados de forma densa, permitindo mapeamentos mais complexos que a simples relação linear possível com o modelo de McCulloch e Pitts (1990). A função de saída y é então mapeada pela aplicação sequencial da Equação 2.1, de modo que a saída da camada N é utilizada na entrada da camada $N + 1$, em processo conhecido como propagação direta (Figura 3a).

Conhecidas as saídas y da última camada, tem-se então a predição do modelo do valor alvo t dada a entrada x. Ao comparar a similaridade entre a saída esperada t e a saída prevista y por uma função de custo, é possível ajustar os pesos w de modo que a diferença entre y e t seja cada vez menor. Dessa forma, define-se um problema de otimização em que os parâmetros w devem ser ajustados na busca de reduzir a função de custo.

A fim de atualizar os pesos da rede de modo adequado, calcula-se o gradiente da função custo, que indica a contribuição de cada peso para o erro na predição. Assim, ajusta-se os pesos por um pequeno incremento na direção oposta ao gradiente, a fim de mover-se na direção contrária a maximização da função custo, i.e. reduzindo o erro gerado pelo sistema. A metodologia de atualização dos pesos é definida pelo otimizador

utilizado, que buscará a solução de mínimo no hiperplano definido pela função de custo. Este processo conhecido como retro-propagação é apresentado na Figura 3b.

2.2 As bases do aprendizado

2.2.1 Função de Ativação

Para determinar a saída y de um dado neurônio, é realizada a combinação linear de suas entradas e seu viés, como discutido previamente na Equação 2.1. Este valor é então utilizado pela função de ativação do neurônio, realizando um mapeamento f para a saída.

A ideia é inicialmente inspirada pelo fenômeno biológico do limiar de ativação, sendo posteriormente corroborada pela constatação de sua importância para possibilitar mapeamentos não-lineares (KARPATHY, 2018). De fato, a utilização da função de ativação trivial linear, ou identidade, não permite a definição de contornos de decisão não-lineares, inviabilizando sua utilização prática.

Tem-se tal comportamento não-linear desejado nas funções logística,

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}}, \tag{2.2}$$

e tangente hiperbólica,

$$\tanh(z) = 2\sigma(2z) - 1, \tag{2.3}$$

que comprimem um número real em intervalos [0, 1] e [-1, 1], respectivamente. Em termos práticos, estas funções apresentam problemas de decaimento do gradiente, de modo que para valores de entrada muito distantes de 0 o gradiente é muito pequeno, dificultando mudanças significativas dos pesos.

Com a utilização da função linear retificada (ReLU),

$$\mathrm{relu}(z) = \max(0, z), \tag{2.4}$$

tem-se a conservação do gradiente para parte significativa dos valores. Há entretanto o risco de cair na zona retificada, efetivamente inutilizando o neurônio. Ainda assim, é uma das funções mais utilizadas, com baixo custo computacional e ótimos resultados empíricos (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Além disso, utiliza-se para problemas de classificação multi classe a função softmax,

$$\mathrm{softmax}(z) = \frac{\mathrm{e}^{z}}{\sum_i \mathrm{e}^{z_i}}, \tag{2.5}$$

na camada de saída, indicando a dependência entre classes, mutualmente exclusivas (ALPAYDIN, 2014). Recomenda-se também a utilização da Equação 2.2 para problemas de classificação binária.

2.2.2 Função Custo

As funções custo são utilizadas para quantificar o prejuízo de se estar em desacordo com a saída desejada, ou seja, permitem avaliar a qualidade dos parâmetros w na busca do comportamento de saída almejado. É uma função que, dada a predição y realizada e o valor que se espera de fato t, calcula o erro associado segundo a métrica escolhida. Neste trabalho foram utilizadas duas funções custo para os treinamentos, sendo estas o erro quadrático médio (MSE),

$$L = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2, \tag{2.6}$$

e entropia cruzada, oriunda do campo da teoria da informação,

$$L = -\frac{1}{N}\sum_{i=1}^{N}\Big(t_i log(y_i) + (1 - t_i) log(1 - y_i)\Big). \tag{2.7}$$

Estas funções definem então a topologia do hiperplano de otimização de alta dimensionalidade[1], de modo que interferem significativamente na performance do otimizador escolhido.

2.2.3 Otimizadores

Otimizadores consistem em algoritmos utilizados para minimizar a função de custo, por meio do ajuste dos parâmetros w que satisfazem esta condição. Existem diversas estratégias de otimização não-convexa[2], sendo os métodos de gradiente os mais utilizados no campo do aprendizado de máquina. Existem basicamente duas metodologias possíveis para o cálculo de gradientes: métodos numéricos e métodos analíticos. O primeiro consiste em uma aproximação numérica por diferenças finitas e é extremamente custoso para se obter, enquanto o segundo utiliza-se do cálculo analítico, sendo exato e eficiente.

A fim de obter os gradientes de modo analítico, utiliza-se do algoritmo de retropropagação, permitindo o cálculo recursivo do gradiente por meio da regra da cadeia. O procedimento é dado pelo cálculo analítico prévio da derivada das funções de ativação, de modo que a atualização δ na saída é ponderada pela taxa de variação do custo pela saída

[1] A função custo pode ser interpretada como um hiperplano cujo mínimo é procurado durante processo de otimização. Pesos da rede neural correspondem a parâmetros que interfere na topologia deste hiperplano, compondo suas dimensões.

[2] Problemas de otimização convexa apresentam solução ótima única, enquanto superfícies não convexas podem apresentam complexas estruturas com diversos pontos de mínimo local.

$$\delta^L = \frac{\partial L}{\partial z^L} = \frac{\partial L}{\partial y^L} f'(z^L), \tag{2.8}$$

enquanto nas camadas internas tem-se

$$\delta^l = \frac{\partial L}{\partial z^l} = \frac{\partial L}{\partial y^l} f'(z^l), \tag{2.9}$$

em que o termo

$$\frac{\partial L}{\partial y^l} = \sum_{i,j} w^l \frac{\partial L}{\partial z^{l+1}} \tag{2.10}$$

reflete a ponderação dos pesos com relação a taxa de variação do custo. Conhecidas as atualizações δ, é possível calcular a taxa de variação do custo em função de qualquer peso da rede, de modo a permitir que os otimizadores atualizem os parâmetros da rede de modo adequado. Alguns dos otimizadores mais utilizados, e testados no escopo deste trabalho, são Adam (KINGMA; BA, 2014), e *Stochastic Gradient Descent* (SGD) (ROBBINS; MONRO, 1951),

$$w^l[t+1] = w^l[t] - \eta \frac{\partial L}{\partial w^l[t]}. \tag{2.11}$$

O termo η da Equação 2.11 é conhecido como taxa de aprendizado, responsável por escalar o vetor gradiente. Assim, define-se o passo de atualização na direção contrária ao gradiente, controlando o comportamento da minimização da função custo. Taxas de aprendizado muito reduzidas incorrem em pequenas atualizações, gerando dificuldades na otimização devido a mínimos locais e tempo excessivo de treinamento. Já taxas muito elevadas podem promover *overshooting*, de modo que o processo de otimização por gradientes pode divergir. A fim de calcular o vetor gradiente, utiliza-se o valor de δ obtido previamente por meio da regra da cadeia

$$\frac{\partial L}{\partial w^l} = a^{l-1} \delta^l. \tag{2.12}$$

2.2.4 Treinamento, Validação e Teste

O processo de treinamento é onde se utiliza o conjunto de treinamento dos dados disponíveis no *data set* de forma a ajustar os pesos da rede na tentativa de melhorar seu desempenho na tarefa que se deseja que esta execute. O treinamento de uma rede qualquer é dividido em épocas e cada época constitui a passagem por todos os elementos do conjunto de treinamento ou um subconjunto aleatório deste com tamanho pré-definido.

Para evitar o ajuste de pesos demasiadamente enviesado para uma única classe, é comum a utilização de *batches* de forma que os pesos só são alterados após observados alguns exemplos e não com cada exemplo apresentado.

A validação do modelo é feita utilizando um conjunto de validação, conjunto sem interseção com o conjunto de treinamento, dos exemplos disponíveis. O processo em si consiste na avaliação do desempenho da rede nesse conjunto de dados em que a rede não foi treinada de forma a simular o comportamento da rede em casos mais gerais. Esta avaliação costuma ser realizada após cada época de forma a se observar o comportamento da rede ao longo do treinamento e escolher o modelo que obteve resultados melhores neste conjunto.

O teste da rede consiste essencialmente da mesma avaliação realizada na validação, com as diferenças principais residindo no fato de que, comumente, o conjunto de teste é diferente do conjunto de validação e o teste é realizado somente após serem escolhidos os pesos da rede. Isso evita que a avaliação dos melhores pesos para aquele conjunto específico seja escolhido, mitigando o efeito de especificidade.[3]

2.2.5 Curvas de Aprendizado

O estudo das curvas de aprendizado é de fundamental importância para a avaliação do comportamento de modelos de aprendizado de máquina, permitindo a identificação de problemas de generalização. Realiza-se pela análise do custo e acurácia (ou erro)[4] ao longo do processo de treinamento nos conjuntos de treino e teste, o diagnóstico de processos conhecidos como *overfitting* e *underfitting*.

Estes processos ocorrem quando o sistema apresenta complexidade de ajuste desbalanceada em relação aos dados, de modo a apresentar mapeamentos muito específicos ao conjunto de treino, *overfitting*, ou mostrando-se incapaz de aprender com os dados de treinamento, *underfitting*. Este é conhecido como o dilema viés-variância e é presente para qualquer sistema de aprendizado de máquina (ALPAYDIN, 2014).

[3] Neste trabalho é utilizado o mesmo conjunto para testes e validação. Entretanto, destaca-se que no conjunto de teste realiza-se a média das classificações ao longo do vídeo, em contraposição a classificação baseada apenas em uma amostra aleatória de quadros realizada no conjunto de validação.

[4] Acurácia é a métrica de quantidade relativa de classificações corretas que o modelo obteve, já a taxa de erro é igual a seu complemento, i.e. 1 − acurácia.

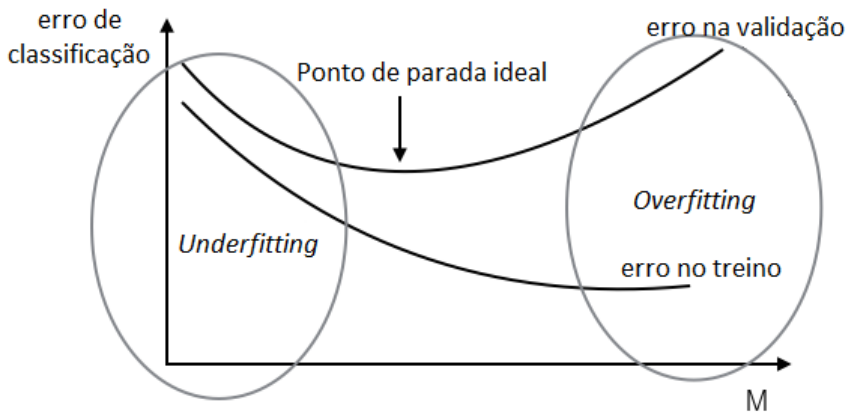


Figura 4 – Curvas de aprendizado. Adaptado de (WEINBERGER, 2015).

De modo prático, pode-se observar na Figura 4 a taxa de erros ao longo de M épocas. No início do processo de treino, observa-se um alto viés, de modo que o regime de *underfitting* é caracterizado pelo alto índice de erros no conjunto de treino e validação. Ao longo do progresso do treinamento, os pesos w do modelo ajustam-se ao conjunto de testes, reduzindo o erro. Entretanto, após um limiar de épocas, verifica-se a transição ao regime de *overfitting*, em que o erro de treino é consideravelmente inferior ao de validação, i.e. o modelo não generaliza em dados nunca vistos.

O estado em que as taxas de erro no treino e validação são de ordem similar, e próximas ao mínimo da validação, é comumente definido como o ponto ótimo de treino do modelo. Tal análise permite não apenas identificar o número de épocas apropriado, mas ajustar adequadamente diversas variações disponíveis no processo de experimentação, como taxa de aprendizado, número de pesos, regularizações entre outros hiperparâmetros.[5]

2.3 *Deep Learning*

2.3.1 Redes Convolucionais

Arquiteturas totalmente conectadas, ou densas, apresentam alto custo computacional, visto que cada neurônio de uma camada conecta-se a todos os neurônios da camada anterior. Desta forma, para sinais de entrada muito grandes, é evidente que este tipo de arquitetura não é razoavelmente escalável. Redes convolucionais (*convolutional neural networks*, CNNs), por outro lado, exploram propriedades de sinais naturais a fim

[5] Hiperparâmetros consistem em variáveis de ajuste definidas para a realização do treinamento do modelo, como número de épocas, tamanho do *batch*, taxa de aprendizado, entre outros.

de otimizar o uso de parâmetros, melhorando a escalabilidade e poder de generalização (KARPATHY, 2018).

Este tipo de rede é composta por basicamente dois estágios, sendo o primeiro composto por aplicações sequenciais de camadas de convolução e *pooling*, responsáveis pela extração automática de características relevantes do sinal de entrada, e o segundo por neurônios densos, os quais finalizam o processo com o mapeamento não-linear das características latentes — de tamanho muito reduzido em relação ao sinal de entrada — extraídas pelas convoluções para a função objetivo.

Camadas convolucionais apresentam conexão local, i.e. cada neurônio convolucional é conectado a apenas um subconjunto de neurônios da camada anterior, pesos compartilhados e independência da quantidade de parâmetros em relação ao tamanho de sua entrada, de modo a permitir profundidade de arquitetura muito superior em relação a redes densas. De fato, tal esquema de conexão neural é melhor compreendido sob ótica da operação de convolução, que nomeia a camada.

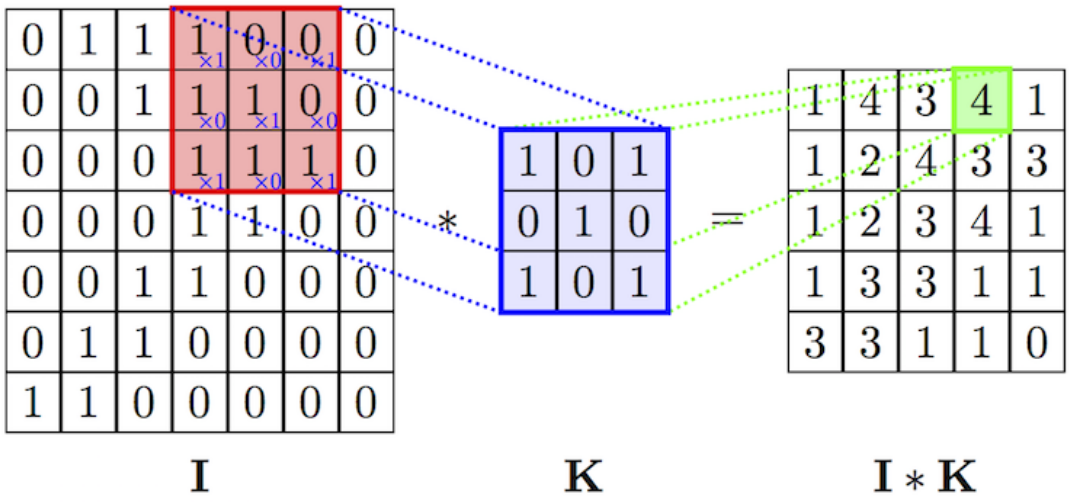


Figura 5 – Convolução 2D. Por (VO, 2018).

A convolução espacial, Figura 5, é uma operação matemática definida em sua forma discreta

$$(I * K)(c1, c2) = \sum_{a1,a2} I(a1, a2) \cdot K(c1 - a1, c2 - a2), \tag{2.13}$$

em que K representa o *kernel*, ou filtro, I a matriz (imagem) a ser convoluída. Pode-se compreender a Equação 2.13 como a passagem de um *kernel* sobre uma imagem, multiplicando seus valores e somando (OLAH, 2014). Cada possível *kernel* K, por sua vez, apresenta propriedades distintas, extraindo diferentes características da imagem. Após a operação de convolução, aplica-se então uma ativação não-linear.

Em camadas convolucionais, o tamanho do filtro define o campo receptivo de cada neurônio, ou seja, a janela de conexão com a camada anterior. Ademais, de cada *kernel* obtém-se a propriedade de compartilhamento de pesos de neurônios, compondo saídas conhecidas como mapa de características, ou *feature maps*, de cada aplicação da Equação 2.13.

Deste modo, cada camada convolucional apresenta N *kernels*, determinando o número de canais que irão compor sua saída. Para determinar as dimensões de altura e largura da saída pode-se utilizar a relação

$$A = \frac{m - f + 2p}{s} + 1, \tag{2.14}$$

em que A corresponde ao tamanho da dimensão na saída, m ao tamanho na entrada, f tamanho do filtro, p tamanho do *padding* e s o passo do filtro.

Observa-se que a aplicação sucessiva destas camadas é capaz de gerar representações hierárquicas do sinal de entrada, de modo que características de alto nível são compostas por aquelas mais baixas (LECUN; BENGIO; HINTON, 2015). Em trabalhos anteriores, foi constatada a detecção de bordas em camadas iniciais, enquanto níveis subsequentes aprenderam a representar objetos ou partes deles (OLAH; MORDVINTSEV; SCHUBERT, 2017).

2.3.2 *Pooling*

Dadas as características locais detectadas pelas camadas convolucionais, utiliza-se uma camada posterior de compressão, unindo características similares em apenas uma e adicionando pequena invariância a posição (LECUN; BENGIO; HINTON, 2015). Esta camada é conhecida como *pooling* em que decidido o tamanho do filtro, este é passado por blocos da imagem e realiza-se uma de duas operações.

1. Mantém o máximo da região descartando os outros elementos, *max pooling*, exemplificado na Figura 6

2. Realiza a média da região, *average pooling*

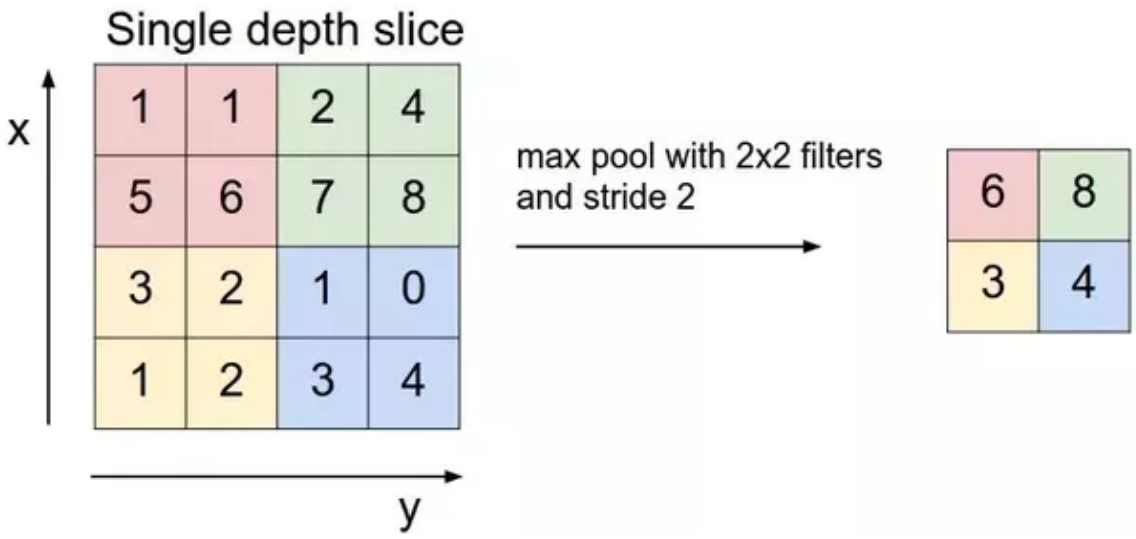


Figura 6 – *Max pooling* de um filtro 2×2 e passo 2. Por (KARPATHY, 2018).

2.3.2.1 *Autoencoders*

Autoencoders consistem em redes que buscam que sua saída seja idêntica a entrada, comprimindo a informação em suas camadas intermediárias. No caso de imagens, fazem uso do processo convolutivo já mencionado para comprimi-las em espaço latente. A ideia é ter uma representação interna da imagem original adequada e utilizar esta representação para armazenar a informação da imagem, diminuindo o espaço gasto para armazenamento da mesma. Também é composta de etapas de de-convolução que se encarregam de recuperar a imagem original a partir da representação em espaço latente.

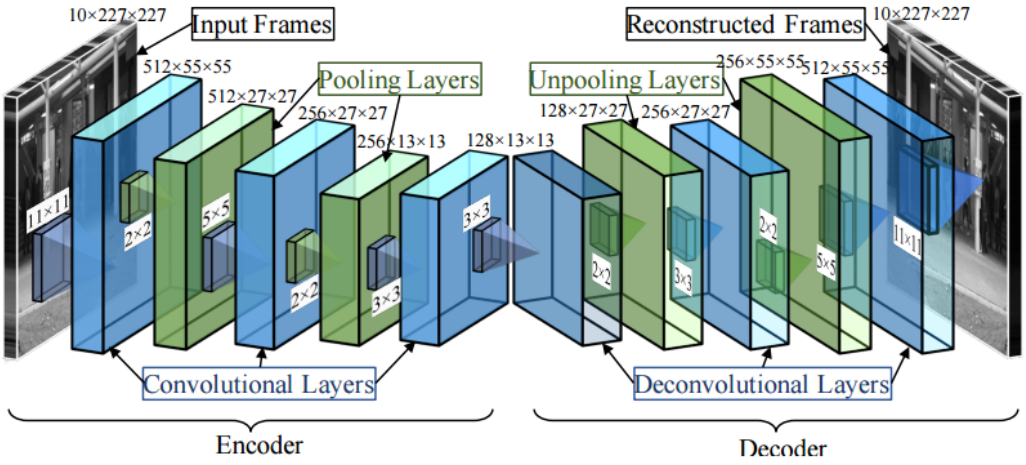


Figura 7 – Estrutura de um *autoencoder*. Por (MANDAL, 2018).

Na figura 7 pode-se observar a estrutura de *autoencoders*. É importante frisar que este tipo de rede não é utilizado diretamente para classificação e que as métricas para avaliar o funcionamento adequado das mesmas são, portanto, distintas, concentrando-se

na capacidade de compressão e qualidade da imagem recuperada. Por estes motivos este tipo de rede foi pouco explorada no contexto deste trabalho.

2.3.3 Redes Recorrentes

Camadas recorrentes possuem semelhanças com as camadas totalmente conectadas. Entretanto, em vez de se conectar diretamente a todas as saídas da camada anterior, a conexão é feita de forma temporal. O módulo recorrente conecta-se a cada bloco temporal e propaga a informação tratada ao longo do tempo, podendo gerar saídas para a próxima camada apenas na última etapa temporal ou em múltiplas etapas.

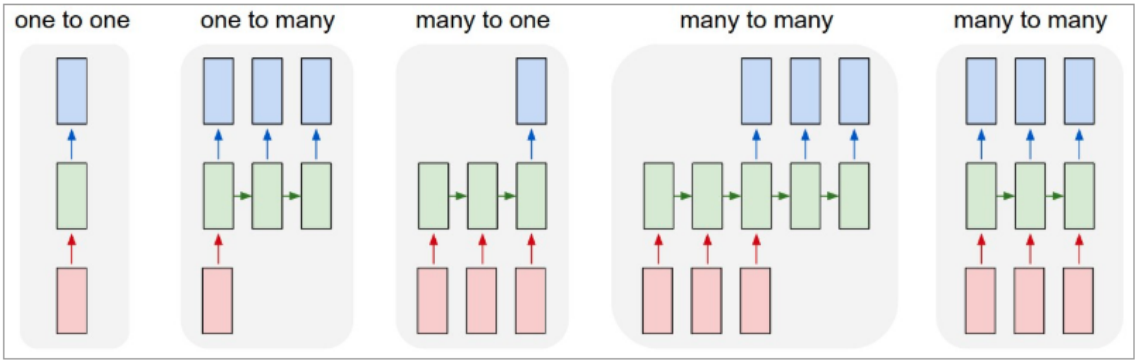


Figura 8 – Modelos Recorrentes. Por (BANERJEE, 2018).

Na Figura 8 podem ser observadas diversas formas de tratamento de dados deste tipo de camada, onde blocos vermelhos correspondem a entradas, blocos verdes ao módulo recorrente e azuis a saídas. A distribuição temporal das informações está representada pela disposição horizontal dos blocos e as conexões laterais presentes na camada recorrente são as informações processadas pelo bloco atrasadas no tempo, de forma a transmitir as mesmas. O treinamento deste tipo de camada ajusta seus parâmetros de forma semelhante ao processo dado nas camadas totalmente conectadas, com o ajuste adicional dos parâmetros de transmissão da informação temporal dentro da própria camada.

2.4 Técnicas de Treinamento

2.4.1 *Transfer Learning*

Uma maneira de acelerar o processo de aprendizagem da rede e reduzir problemas de *overfitting* é a utilização de *transfer learning*. Este processo consiste em aproveitar uma rede previamente treinada em algum problema e aplicá-la, ou a maior parte dela, em uma situação diferente.

O aspecto de redução no tempo necessário para a aprendizagem é relativamente claro, uma vez que a rede já aprendeu a tratar alguns dados. O treinamento restante

já tem proveito destes pesos e não precisa ajustá-los tanto quanto precisaria com uma inicialização aleatória.

Por outro lado, a redução do *overfitting* se dá pelo fato de a rede ter ajustado seus pesos não só nos exemplos do problema em questão, mas também no *data set* em que foi utilizada anteriormente. Com isto, evita-se que a rede aprenda a representar apenas especifidades dos dados de treinamento atual, visto que ela já tem conhecimento de representação de dados diferentes.

2.4.2 *Data Augmentation*

Na tentativa de aumentar a variedade de exemplos disponível, foram realizadas transformações aleatórias nos dados de treinamento, aumentando a robustez do modelo a variações. Esse processo é conhecido como *data augmentation*, comumente utilizado por reduzir efeitos de *overfitting* e aumentar o conjunto de treino de modo barato, apesar da alta correlação das imagens geradas com as originais.

Utilizou-se como base as técnicas de transformação em imagens disponibilizadas por Jung (2018). São práticas comuns a aplicação de rotação, de ruído, embasamento, espelhamento, sombreamento, mudança de escala, distorção e, recentemente, oclusão aleatória (ZHONG et al., 2017). No caso específico de vídeos, é necessário cautela, já que transformações realizadas em cada quadro devem ser consistentes temporalmente. Além disso, possibilita-se realizar variações temporais — e não somente espaciais — ao selecionar recortes em diferentes momentos do vídeo.

Para a adaptação destas técnicas, observou-se que as transformações realizadas são independentes da quantidade de canais e da correspondência dos mesmos. Portanto realizando um arranjo diferente dos quadros do vídeo é possível utilizar diretamente as técnicas desenvolvidas para transformações em imagens. Deve-se tomar cuidado para manter as disposições originais de linhas e colunas e reverter o arranjo após as transformações.

Como este é um processamento extra dos dados, requer tempo para ser executado. É suficientemente rápido para se realizar em meio ao treinamento, mas armazenar dados alternativos após as transformações é uma opção para acelerar o processo. Optando-se por esta alternativa, deve-se tomar o cuidado de selecionar aleatoriamente dentre os dados transformados e originais, aumentando a variedade do treinamento.

2.4.3 Regularização

Um problema já comentado a respeito de redes neurais é a possibilidade de se ter *overfitting*. Existem algumas técnicas para tentar evitar esta ocorrência, fazendo com que o modelo tenha maior capacidade de generalização do problema.

Uma técnica muito utilizada é o *dropout*, onde, durante o treinamento, alguns neurônios são inutilizados de forma aleatória. Desta forma a rede precisa se adaptar a tentar conseguir um bom resultado utilizando menos parâmetros, esta aleatoriedade acaba por simular o treinamento de mais redes de tamanho menor. Fazendo-se isto, aumenta-se a robustez da rede, tornando-a mais capaz de generalização.

Outras técnicas utilizadas são as regularizações L_1 e L_2, nas quais procura-se evitar pesos de grande magnitude, reduzindo a dependência da rede a características muito específicas. Este processo é realizado adicionando as normas dos pesos, ponderadas por um fator λ, à função de custo, nas formas referentes a regularização L_1

$$C = L + \lambda \sum ||w||, \tag{2.15}$$

e a L_2

$$C = L + \lambda \sum ||w||^2, \tag{2.16}$$

em que C é o custo total, L é a quantificação do custo pelo erro, w são os pesos, λ é um parâmetro livre e define o grau de penalização em relação a normas muito elevadas.

2.4.4 Aprendizagem Cíclica

Decidir os hiperparâmetros da rede é tarefa fundamental e não-trivial. Dentre estes, destaca-se a importância de definir adequadamente a taxa de aprendizagem do modelo. Apresenta-se a seguir considerações a respeito da definição da taxa de aprendizado por meio de técnica de aprendizagem cíclica, ou *one cycle* (SMITH, 2018).

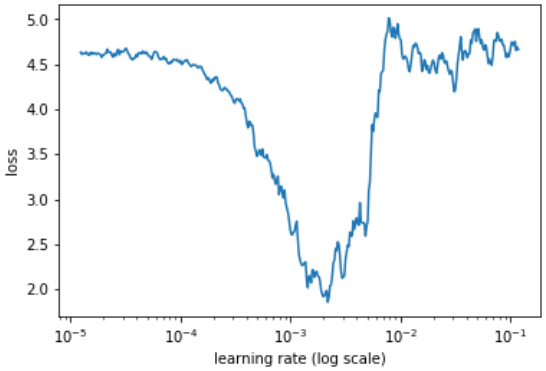


Figura 9 – Comportamento do custo em função da taxa de aprendizado.

Um método possível, e que requer um longo tempo, para definir a melhor taxa de aprendizado η, é treinar a rede com múltiplos valores η e observar qual apresenta os melhores resultados. Outra opção é realizar os testes dentro de uma época e observar o comportamento da função de custo para diferentes taxas. Analisada esta relação, como

ilustrado na Figura 9, é possível ter uma noção da curva de custo em função da taxa e escolher um η adequado.

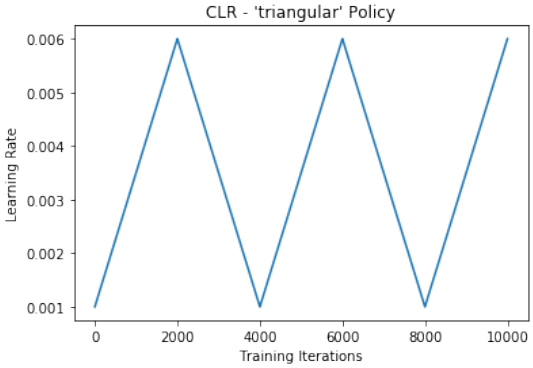


Figura 10 – *One cycle*. Por (KENSTLER, 2016)

De forma geral é interessante utilizar a maior taxa em que o custo não começou a subir, pois esta evita mínimos locais mais facilmente e o desempenho da rede melhora mais rapidamente. Entretanto, existem vantagens em se iniciar o treinamento com uma taxa menor e aumentá-la posteriormente. Um procedimento interessante é o de *one cycle*, Figura 10, onde se inicia o treinamento com uma taxa mais baixa, aumentando a taxa linearmente até a metade do treinamento e depois torna-se a abaixá-la.

Esta metodologia apresenta um progresso extremamente rápido e o processo ainda ajuda a evitar a ocorrência de *overfitting*. É extremamente útil para economizar tempo e evita testes com muitas taxas diferentes, contribuindo para busca de otimização de outros parâmetros.

3 Trabalhos Relacionados à Classificação de Vídeos

3.1 Visão Geral

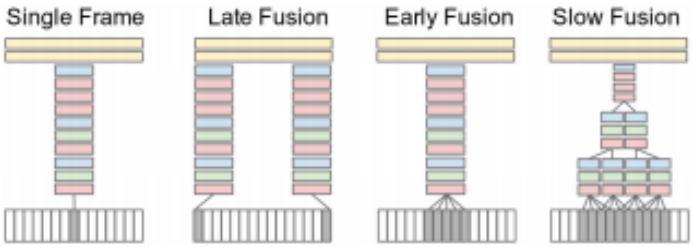


Figura 11 – Agrupamentos Diversos. Por (KARPATHY et al., 2014).

Explora-se em trabalho inicial de Karpathy et al. (2014) a possibilidade de utilizar agrupamentos de *frames* de um filme por meio de camadas convolucionais, observando o efeito de processar as informações distribuídas no domínio temporal sob diferentes arquiteturas. A entrada para a rede é variada de forma a acomodar quantidades diversas de quadros nas camadas convolucionais, realizando a combinação da informação temporal de múltiplas maneiras. Na Figura 11 são apresentadas formas propostas para esta análise, onde blocos vermelhos indicam camadas convolucionais, verdes camadas de *pooling*, azuis camadas de normalização e amarelos camadas totalmente conectadas.

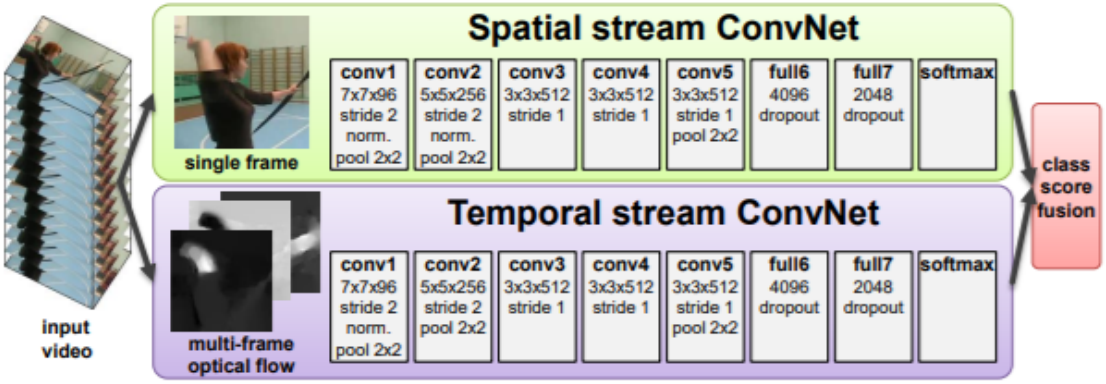


Figura 12 – Arquitetura com representação dupla. Por (SIMONYAN; ZISSERMAN, 2014a).

Em implementação proposta por Simonyan e Zisserman (2014a), Figura 12, para aproveitar as características temporais de vídeos, observa-se a utilização de duas representações desta informação: imagem natural e fluxo óptico. Realizada esta separação, as

duas entradas são processadas por redes paralelas, de arquitetura muito semelhante, e suas classificações individuais são combinadas para gerar a saída final da rede.

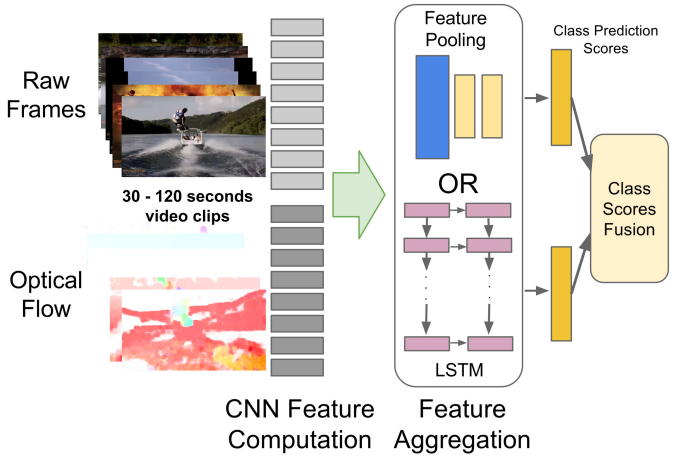


Figura 13 – Agrupamentos alternativos. Por (NG et al., 2015).

Uma abordagem diferente para este problema é dada por Ng et al. (2015) em que se explora possibilidades de agrupamento de informações temporais após o processamento convolucional de imagens. O procedimento proposto faz uso de quadros do vídeo e fluxo óptico local como dados para entrada inicial da rede. A arquitetura é muito similar a apresentada por Simonyan e Zisserman (2014a), utilizando, no entanto, CNNs mais modernas como AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e GoogLeNet (SZEGEDY et al., 2014).

Além disso, verifica-se novas soluções para considerar características globais, com a utilização de camadas de *pooling* ao final da rede aplicadas à integra do vídeo processado. A segunda metodologia apresentada dá-se com uso de redes recorrentes do tipo LSTM (HOCHREITER; SCHMIDHUBER, 1997) em substituição às camadas de *pooling* supracitadas. As duas variações podem ser observadas na Figura 13.

É proposto por Silva (SILVA, 2017) a utilização de histogramas de classificações locais para determinar a classificação global do vídeo. Esta metodologia aborda a informação global do vídeo como um empilhamento de informações locais. Ao se classificar diversas vezes o vídeo de acordo com características no entorno das regiões, é esperado que o maior número de indicações seja dado à categoria adequada do vídeo. É realizada a dupla representação proposta por Simonyan e Zisserman (2014a) e gerado um histograma para cada representação.

Outras variações de modelos de duas representações em paralelo são discutidas. A fim de eliminar a etapa de pré-processamento de fluxo óptico — gerando aumento do armazenamento de dados e custo prévio ao modelo —, Zhu et al. (2017) propõe a utilização de um codificador de movimento interno à própria rede. Nova possível variação

consiste na fusão das características internas extraídas dos fluxos naturais e óptico para uma camada densa, em contraposição às abordagens de combinação das predições.

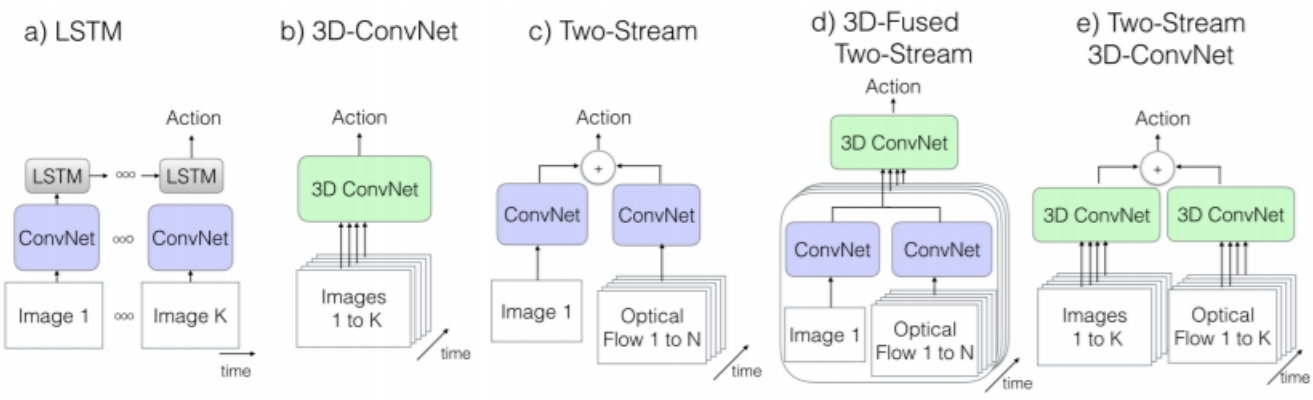


Figura 14 – Arquiteturas padrão em classificação de vídeo. Por (CARREIRA; ZISSERMAN, 2017b).

Além das técnicas supracitadas de codificação temporal — como redes recorrentes e agrupamento de *frames* —, tem-se como alternativa a utilização de camadas convolucionais tridimensionais (TRAN et al., 2015). Dessa forma, é possível não apenas aprender filtros capazes de extrair informações espaciais relevantes de um único quadro do vídeo, mas obter concomitantemente relações temporais em um único procedimento, i.e. representar características espaço-temporais. Trabalho posterior de Carreira e Zisserman (2017b) avança os limites de redes convolucionais 3D, definindo o atual estado da arte na área. A Figura 14 sintetiza algumas das técnicas mais utilizadas atualmente para classificação de vídeo.

Apesar da dificuldade em obter de redes convolucionais 3D pré-treinadas em um grande banco de vídeos, tal como tem-se com o ImageNet (DENG W. DONG; FEI-FEI, 2009), é possível utilizar técnicas de transferência de conhecimento para melhorar consideravelmente a performance dos modelos. Para tal, realiza-se o *trasnfer learning* supervisionado por uma rede convolucional 2D para a rede não treinada 3D (DIBA et al., 2017), permitindo a utilização de vídeos não anotados para o treinamento do modelo.

3.2 Arquiteturas

3.2.1 Quadro Único

Nesta seção são apresentados modelos que abordam a classificação de vídeo utilizando apenas uma imagem estática retirada do mesmo. Pode-se então realizar a predição do vídeo utilizando todos os quadros, observando a média de ativação da última camada por quadro. Entretanto, esta abordagem ainda não considera a relação temporal entre os *frames*, constituindo-se em modelos *baseline* para a exploração de arquiteturas de classi-

ficação de vídeo. A seguir, descreve-se as arquiteturas VGG (SIMONYAN; ZISSERMAN, 2014b), ResNet (HE et al., 2015) e Inception (SZEGEDY et al., 2014) investigadas no contexto da classificação de vídeo.

3.2.1.1 VGG

A arquitetura VGG segue o caminho definido pela rede AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), ao utilizar camadas convolucionais para solucionar o *data set* ImageNet (DENG W. DONG; FEIFEI, 2009), em que se busca classificar corretamente elementos de uma imagem. Tal *data set* é composto de 1000 categorias com aproximadamente $1,2$ milhões de exemplos de treinamento, 50000 exemplos de validação e 100000 de teste, em que as categorias oficiais de cada exemplo foram determinadas por análise humana. De fato, esta abordagem convolutiva mostrou-se significativamente superior às técnicas de redes densas utilizadas anteriormente, consagrando a superioridade da arquitetura CNN no problema de classificação de imagens.

Tem-se na VGG a utilização de técnicas propostas previamente por Krizhevsky, Sutskever e Hinton (2012), destacando-se o uso da não-linearidade linear retificada (*Rectified Linear Unit*, ReLU) como função de ativação. Desta forma, acelera-se o processo de treinamento e evita-se problemas de regiões de saturação da função de ativação, que favorecem redução significativa da atualização dos pesos (*vanishing gradient problem*). Consolida-se também o uso de técnicas de regularização utilizadas na AlexNet como *data augmentation* e *dropout*.

Apresenta-se como diferencial em relação a arquiteturas anteriores a implementação de campos receptivos de tamanho 3×3 com deslocamento unitário, significativamente menores do que tamanhos 11×11 com passo 4 sugeridos por Krizhevsky, Sutskever e Hinton (2012). Desta forma, opta-se pela utilização de um agrupamento de cinco camadas convolucionais 3×3 a fim de obter campo receptivo análogo a apenas uma camada 11×11. Como vantagem, observa-se a redução significativa no número de parâmetros do modelo, bem como a incorporação de múltiplas não-linearidades adicionais, favorecendo funções de decisão mais discriminativas.

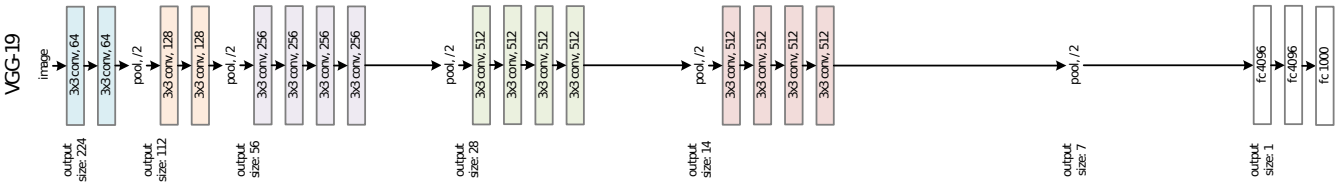


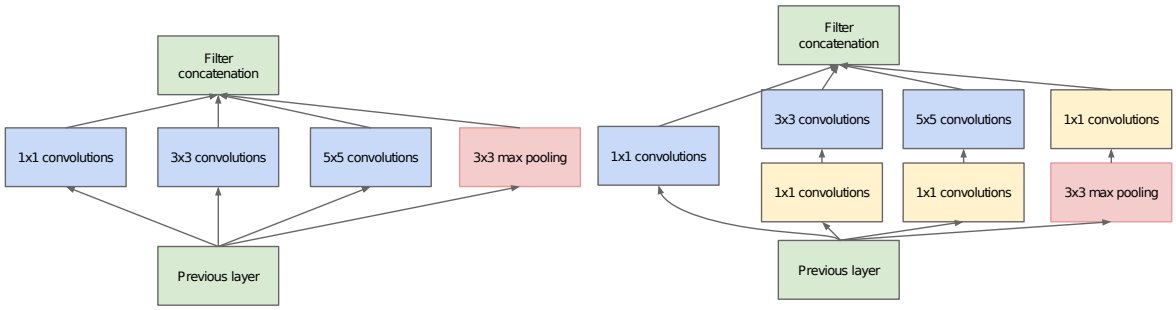
Figura 15 – Arquitetura da VGG19. Por (HE et al., 2015).

A variação VGG19 selecionada para testes apresenta 19 camadas, sendo composta por diversas operações convolucionais seguidas por três redes totalmente conectadas, como

pode ser observado na Figura 15. A fim de aproveitar seu treinamento prévio no ImageNet, substitui-se as camadas totalmente conectadas, mantendo os pesos das camadas convolutivas resultantes do treinamento prévio.

3.2.1.2 Inception

Proposta originalmente em 2014, a arquitetura Inception apresentou resultados estado da arte no *data set* ImageNet do mesmo ano. O modelo parte da ideia de eliminar o processo de busca do tamanho do campo receptivo de cada camada convolucional manualmente, permitindo que o processo de otimização selecione as dimensões mais convenientes para os filtros.



(a) Módulo Inception, versão ingênua

(b) Módulo Inception com redução de dimensionalidade

Figura 16 – Módulo Inception. Por (SZEGEDY et al., 2014)

Para obter tal processo de escolha automática, Szegedy et al. (2014) propõe o módulo Inception, apresentado na Figura 16. Em sua versão ingênua, aplica-se as operações de convolução — com filtros de tamanhos 1×1, 3×3, 5×5 — e *max pooling* — tamanho 3×3 — paralelamente, concatenando as saídas de cada operação, como visto na Figura 16a. Dessa forma, permite-se a captura de características em diferentes escalas, de modo que campos receptivos maiores são capturados a partir de processo análogo ao discutido na Seção 3.2.1.1.

Entretanto, esta abordagem apresenta alto custo computacional, limitando consideravelmente a profundidade viável da rede. Faz-se necessário, pois, realizar otimizações no módulo Inception proposto, utilizando a operação de convolução 1×1 proposta por Lin, Chen e Yan (2013), responsável por condensar o número de filtros de uma camada e adicionar uma não-linearidade ao longo do processo. Dado um volume de entrada de dimensões $N_h \times N_w \times N_{ch_{in}}$ pode-se convoluir com $N_{ch_{out}}$ *kernels* com tamanho $1 \times 1 \times N_{ch_{in}}$ de modo a obter uma saída $N_h \times N_w \times N_{ch_{out}}$.

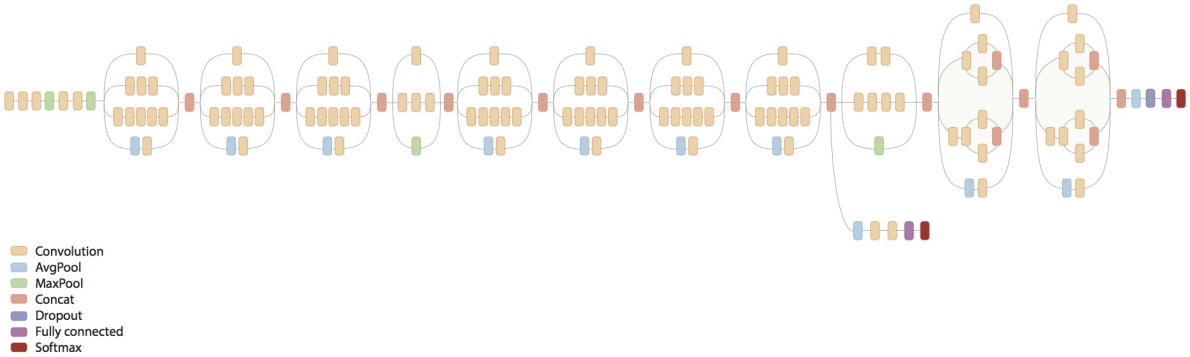


Figura 17 – Arquitetura da InceptionV3. Por (RADEK, 2017).

Compõe-se desta maneira o módulo Inception com redução de dimensionalidade descrito na Figura 16b, que recorre a convoluções 1×1 previamente às custosas convoluções 3×3 e 5×5. A arquitetura é finalmente composta por sequências destes módulos, capazes de realizar complexos mapeamentos sob custo reduzido. Emprega-se neste trabalho a terceira versão deste modelo, InceptionV3 (veja Figura 17), em que Christian et al. (2015) propõe a decomposição de convoluções maiores em múltiplas convoluções reduzidas (e.g. um campo 5×5 em duas operações 3×3), obtendo maior eficiência.

3.2.1.3 ResNet

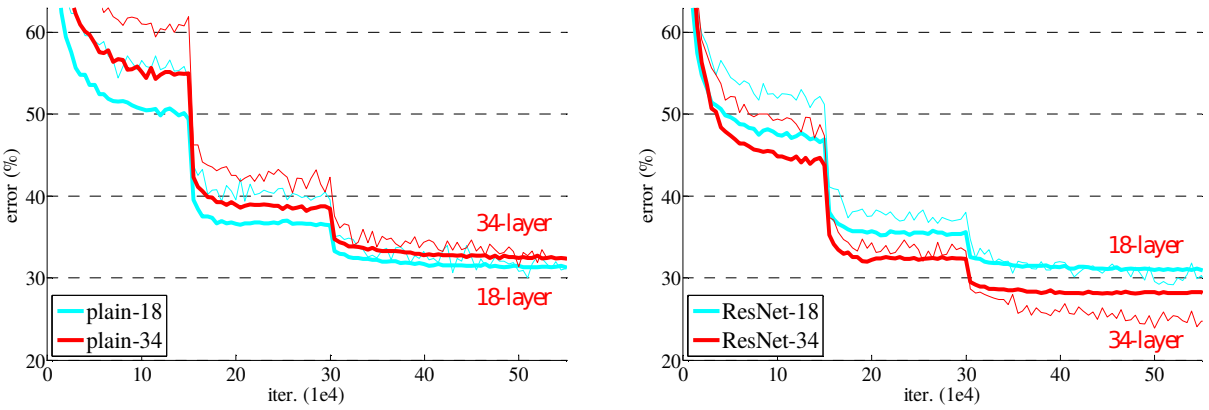


Figura 18 – Treinamento no ImageNet de redes simples em comparação com redes residuais. Por (HE et al., 2015).

Um ano após o sucesso da Inception, a arquitetura ResNet apresentou-se como a nova vencedora do ImageNet. Inspirada na observação de que redes convolucionais mais profundas nem sempre apresentam resultados de treinamento superiores, propôs-se o conceito de módulos residuais que se utilizam de conexões diretas capazes de superar tal problema.

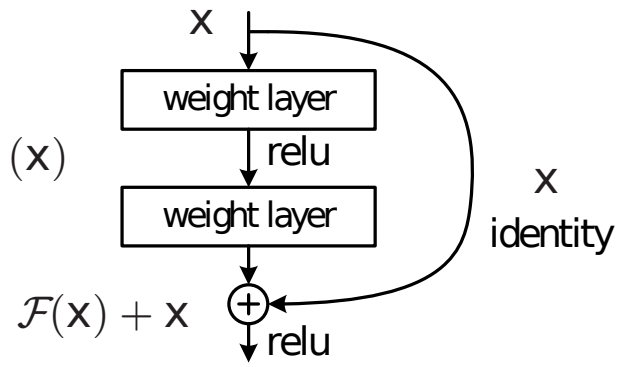


Figura 19 – Módulo Residual. Por (HE et al., 2015).

Como observado na Figura 18, uma arquitetura simples com maior número de camadas pode apresentar resultado inferior a sua versão reduzida. De fato, este fenômeno é contra-intuitivo, visto que a adição de uma camada é teoricamente capaz de promover transformações mais complexas ou, caso necessário, manter o estado anterior, efetuando o mapeamento identidade. A partir deste cenário, He et al. (2015) propõe a adição de uma conexão direta (vide Figura 19), de modo que substituí-se a busca do mapeamento direto $H(x)$ pelo mapeamento residual $F(x) := H(x) - x$. O problema é reformulado, pois, como a solução de $F(x) + x$.

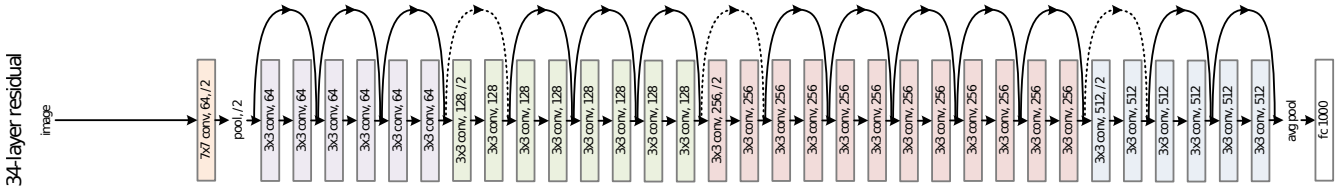


Figura 20 – Rede Residual. Por (HE et al., 2015).

Dessa forma, facilita-se o aprendizado do mapeamento identidade, em que se tem a solução trivial $F(x) = 0$. Além disso, facilita-se a retro-propagação dos erros, evitando que camadas iniciais apresentem problemas de gradientes pequenos, de modo a apresentar atualizações insignificativas em seus pesos. O resultado é verificado na Figura 18, em que a arquitetura residual apresenta performance superior com o incremento da profundidade da rede. A arquitetura final da ResNet é apresentada na Figura 20 em que se apresenta variante com 34 camadas. Utiliza-se no nosso trabalho a arquitetura ResNet50, com 50 camadas.

3.2.2 Múltiplos Quadros

Nesta seção discutem-se modelos que utilizam diversos quadros para efetuar a classificação do vídeo. Dessa forma, explora-se efetivamente a característica temporal do vídeo, modelando a dinâmica entre quadros. Devido a limitações dos modelos, não é possível utilizar todos os *frames* disponíveis, de forma que o processo de inferência de todo o

vídeo é dado pela média das ativações da última camada por blocos de quadros. Assim, tem-se a exploração efetiva das dinâmicas locais, enquanto fenômenos globais são tratados sem efetiva relação temporal. A seguir, descreve-se as arquiteturas pesquisadas: convolucional recorrente (NG et al., 2015; DONAHUE et al., 2014) e convolução tridimensional (CARREIRA; ZISSERMAN, 2017b; HARA; KATAOKA; SATOH, 2017; TRAN et al., 2015).

3.2.2.1 Convolucional Recorrente

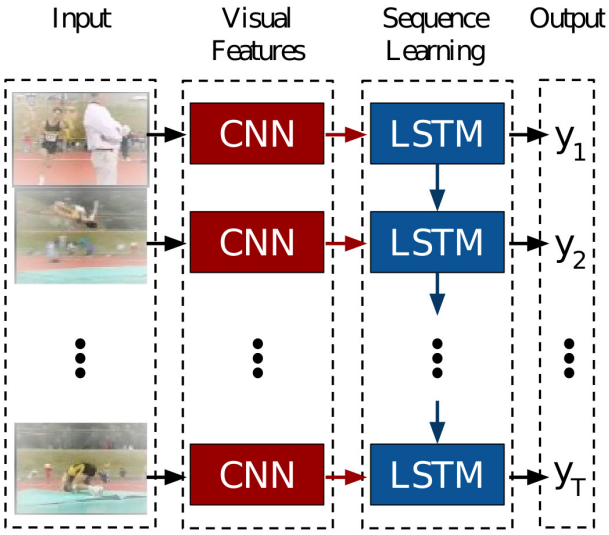


Figura 21 – Arquitetura da rede convolucional recorrente. Por (DONAHUE et al., 2014).

As redes deste tipo são constituídas de camadas convolucionais distribuídas ao longo do tempo para extrair características dos quadros utilizados. Em seguida essas características são interpretadas por uma ou mais camadas recorrentes do tipo LSTM. Por fim é utilizada uma camada totalmente conectada para classificação do vídeo.

Com esta arquitetura, apresentada na Figura 21, tem-se a intenção de modelar a relação temporal entre os quadros por meio da utilização das camadas recorrentes LSTM, visto que a memória delas permite a interação entre os dados sequenciais. Dessa forma, há na CNN a interpretação espacial e na LSTM, a temporal. No escopo deste trabalho, utilizou-se apenas a saída recorrente final, y_T, como característica disponível à rede densa.

3.2.2.2 Convolução Tridimensional

A ideia de se utilizar convoluções 3D para tratar de vídeos em oposição a imagens é um passo lógico e bastante explorado. Entretanto, dada a elevada quantidade de parâmetros, o treinamento de redes tridimensionais profundas apresenta sérios problemas de *overfitting*. Em imagens essa dificuldade é mitigada pelo uso de grandes *data sets* consolidados, contudo a disponibilidade e extensão de conjuntos extensos de qualidade para vídeos ainda é questionável.

Tendo em vista esta problemática, Hara, Kataoka e Satoh (2017) propõe um estudo da viabilidade do uso de arquiteturas profundas CNN 3D. O modelo de rede escolhido para composição das redes convolucionais 3D é a extensão tridimensional da ResNet, tendo em vista o sucesso de sua versão bidimensional.

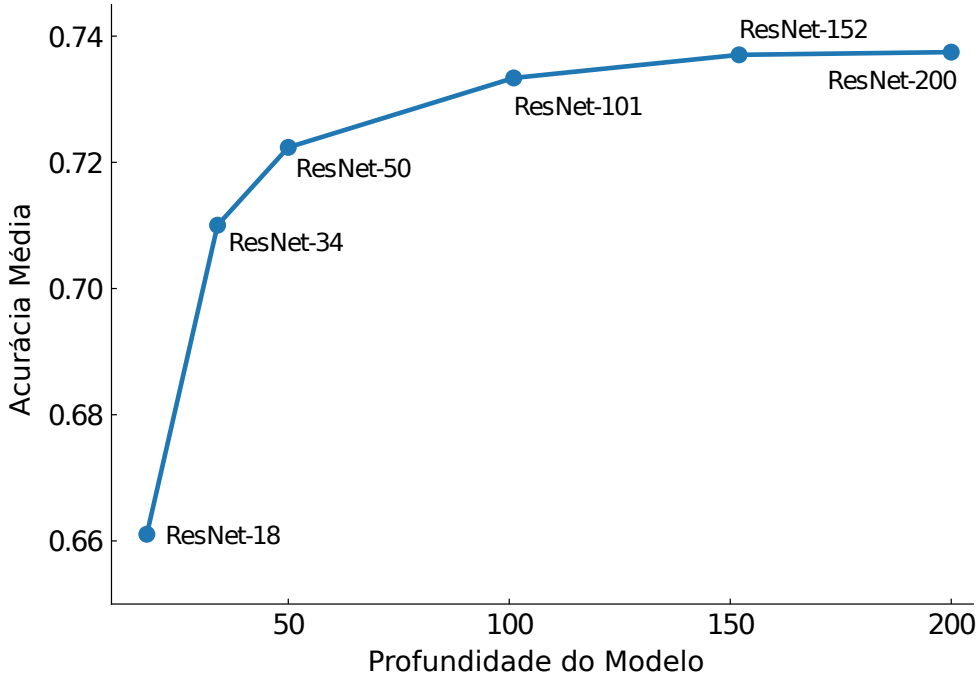


Figura 22 – Relação entre profundidade das redes ResNets 3D e seu desempenho no *data set* Kinetics. Adaptado de (HARA; KATAOKA; SATOH, 2017).

Realiza-se então análise da acurácia do modelo em função de sua profundidade, argumentando-se que caso o conjunto de dados seja extenso o suficiente, comportamento análogo ao apresentado em redes 2D deve ocorrer. Na Figura 22, observa-se que a rede 3D utilizada apresenta a característica de melhorar sua acurácia na validação com a utilização de mais camadas, indicação de que o processo de *overfitting* é bem mitigado no *data set* utilizado, em comportamento análogo ao apresentado em redes 2D.

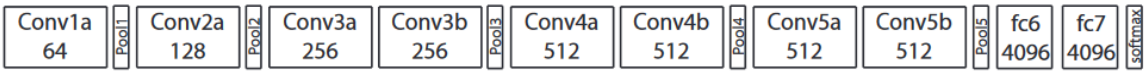


Figura 23 – Arquitetura C3D. Por (TRAN et al., 2015).

Em trabalhos anteriores, propôs-se o uso de arquiteturas compactas, denominadas C3D (TRAN et al., 2015), a qual baseia-se na possibilidade de representação de características espaço-temporais em redes convolucionais tridimensionais. É explorada a relação entre tamanhos dos filtros e comportamento da rede. Concluiu-se que, dentro dos testes realizados, a dimensão mais adequada é de filtros homogêneos $3 \times 3 \times 3$. Sendo assim sua arquitetura final, observada na Figura 23, é composta por tais camadas convolutivas e

max pooling $2 \times 2 \times 2$, excluída a primeira camada de *max pooling* que apresenta tamanho $1 \times 2 \times 2$ com o intuito de se preservar a informação temporal no início do processo.

Em proposta inovadora, define-se a I3D (CARREIRA; ZISSERMAN, 2017b), baseada na arquitetura da InceptionV1. Uma opção destacada para seu treinamento é utilizar os parâmetros encontrados pela versão 2D da InceptionV1, por meio de um processo de *inflação* da rede para a estrutura tridimensional, distribuindo os pesos originais na dimensão temporal.

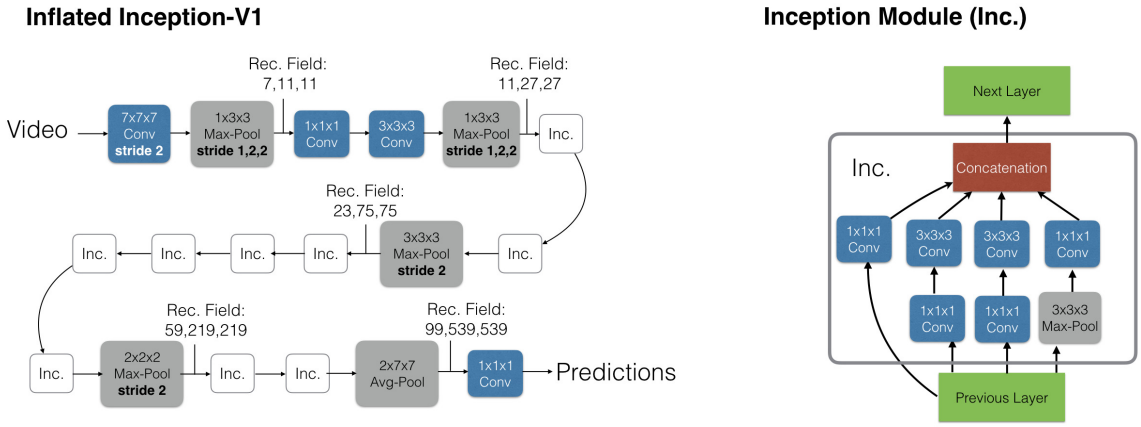


Figura 24 – Arquitetura da I3D. Por (CARREIRA; ZISSERMAN, 2017b).

É apresentado na Figura 24 a arquitetura da I3D, utilizada em treinamento com quadros do vídeo original e fluxo ótico, *optical flow*, treinando-se as redes separadamente. Para classificação final do vídeo faz-se a média das classificações em cada caso. Neste trabalho utilizamos a estrutura proposta para o I3D, entretanto a classificação com esta arquitetura é feita utilizando apenas o vídeo original. Para aproveitar o treinamento realizado, utiliza-se a rede inicializada com os parâmetros encontrados para o *data set* Kinetics (KAY et al., 2017), composto de aproximadamente 500000 clipes de vídeos divididos em 600 categorias de ações humanas. Com os parâmetros obtidos do *transfer learning*, treinamos a rede como um todo no *data set* UCF101.

3.2.3 Motion Flow

Os modelos que fazem uso do *motion flow* estarão descritos nesta seção. Para utilizar esta representação do vídeo, foi realizado o processamento prévio e gravou-se o resultado em disco. Em todos os casos é possível utilizar o vídeo inteiro fazendo-se a média da ativação da última camada em todos os blocos, mas este método não considera relação temporal entre os blocos.

Para tratar o vídeo com *optical flow* faz-se uso de duas torres convolucionais, por isso o modelo será tratado como *two-stream*, em que uma torre atua em um quadro de referência do vídeo original e outra em N quadros, em torno do quadro de referência do *optical flow*. Para acomodar múltiplos quadros em uma convolução 2D, estes são distribuídos nos canais. Ao fim das torres, é realizada a fusão das ativações para se determinar a classificação final do modelo.

O modelo proposto por Simonyan e Zisserman (2014a) e ilustrado na Figura 12 foi uma das abordagens implementadas. Uma alteração testada foi a mudança da torre utilizada na imagem de referência, em que utiliza-se a InceptionV3 em substituição a CNN proposta. Nas etapas de camadas totalmente conectadas e fusão das ativações foram realizadas diversas abordagens, dentre elas tem-se:

- fazer a média das ativações das torres;
- passar a média por uma camada totalmente conectada e
- concatenar as características extraídas por cada torre CNN e fazer a classificação final com algumas camadas totalmente conectadas.

4 Desenvolvimento

4.1 Data sets

Descreve-se a seguir os *data sets* utilizados durante pesquisa de modelos de reconhecimento de vídeo.

4.1.1 UCF101

Figura 25 – Classes do UCF101 *Action Recognition Data Set*. Por (SOOMRO; ZAMIR; SHAH, 2012).

O UCF101 (SOOMRO; ZAMIR; SHAH, 2012) consiste em um *data set* de reconhecimento de ações a partir de vídeos provenientes do *YouTube*, sendo composto de 13320 clipes de vídeo pertencentes a 101 classes distintas. Dentro de cada classe há agrupamento de exemplos que possuem características em comum, como cenário semelhante. A separação dos dados é feita de forma a garantir que não existam exemplos do mesmo grupo no treinamento e no teste.

As possíveis classes de ações são apresentadas na Figura 25. Além disso é possível separar estas classes em 5 categorias:

1. Interação humano-objeto;
2. Movimento corporal apenas;
3. Interação entre humanos;
4. Tocando instrumento musical;
5. Esportes.

São disponibilizadas 3 separações dos exemplos entre conjuntos de treinamento e teste. Visto que exemplos diferentes extraídos de um mesmo vídeo são propositalmente agrupados em um mesmo conjunto, a utilização de um subconjunto de treinamento como validação favorece estimativas muito otimistas em relação a realidade, de modo que impossibilita-se seu uso. Ademais, selecionar um subconjunto de dados de teste como validação favorecem estimativas confiáveis, inviabilizando, porém, a comparação com resultados apresentados na literatura. Dessa forma, seguimos a prática usual adotada a este *data set* ao utilizar o conjunto de teste como validação, com o devido conhecimento e cuidados cabíveis com a prática. Além disso, a utilização da metodologia *k-fold*, neste caso com 3 *folds*, minimiza os problemas dessa abordagem.

4.1.2 HMDB51

O HMDB51 (KUEHNE et al., 2011) é composto de 6766 clipes de vídeo pertencentes a 51 classes distintas.

Assim como ocorre com o UCF101, são disponibilizadas 3 separações dos exemplos entre treinamento e teste de acordo com as características dos exemplos, de modo que a abordagem de utilizar o conjunto de teste como validação é novamente realizada.

As classes podem ser associadas a 5 categorias distintas. A seguir são listadas as classes indicando a que categoria pertencem.

1. Ações faciais gerais: *smile, laugh, chew, talk;*

2. Ações faciais com manipulação de objetos: *smoke, eat, drink;*

3. Movimentos gerais de corpo: *cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, hand-stand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave;*

4. Movimentos de corpo com interação com objeto: *brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw;*

5. Movimentos de corpo para interação com humanos: *fencing, hug, kick someone, kiss, punch, shake hands, sword fight.*

4.2 Aspectos Práticos

4.2.1 Implementação

Os modelos utilizados neste trabalho foram construídos em linguagem de programação Python utilizando as bibliotecas Keras (CHOLLET et al., 2015) e Tensorflow (ABADI et al., 2015). Os modelos convolucionais VGG, ResNet e Inception estão diretamente disponíveis em Keras, incluindo a opção de utilizar pesos previamente ajustados durante treinamento no desafio ImageNet ou inicializados aleatoriamente, necessitando apenas de adaptação nas camadas totalmente conectadas no topo dos modelos. Para os modelos de múltiplos quadros foi necessária a construção das arquiteturas, facilitada pelos recursos disponíveis nas bibliotecas.

No caso das redes convolucionais-recorrentes, a parte convolucional utilizada foi a Inception disponível em Keras, no entanto a parte recorrente foi construída manualmente e inicializada com pesos aleatórios. Para os modelos *two-stream* as torres convolucionais foram montadas seguindo a descrição de Simonyan e Zisserman (2014a) e inicialização aleatória dos pesos, com eventual substituição da torre do quadro de referência pelo modelo Inception. Para a rede convolucional 3D utilizou-se a implementação de Ese (2018) baseada na versão oficial de Carreira e Zisserman (2017a).

As implementações realizadas e demais bibliotecas secundárias utilizadas são disponibilizadas no repositório oficial do projeto (CARVALHO; MARQUES, 2018). Para todo o desenvolvimento do trabalho utilizou-se o seguinte *setup* de *hardware*:

- processador Intel(R) Core(TM) i7-8700 CPU @ 3,20 GHz,
- memória de 16,0 GB e
- placa gráfica NVIDIA GeForce GTX 1080 TI.

4.2.2 Carregamento de Dados

Durante a execução do trabalho, o tratamento de quantidades significativas de dados mostrou-se um dos maiores entraves para a experimentação efetiva com os modelos de *deep learning* pesquisados. De fato, a implementação ingênua inicial consumia aproximadamente 47 minutos por época, sem a utilização de *data augmentation*.

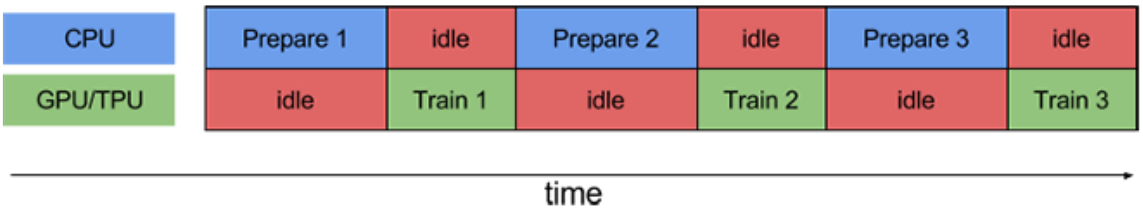


Figura 26 – Implementação ingênua processamento de entrada. Por (ABADI et al., 2015).

Pode-se compreender o processo realizado por meio da Figura 26, na qual verifica-se que em grande parte do tempo subutiliza-se tanto a CPU como a GPU disponíveis. Além disso, decodifica-se o video selecionado até o último quadro desejado, de modo que a maior parte dos quadros iniciais é processada em vão.

Como primeira otimização, transformou-se o *data set* de vídeos em imagens. Assim, evitou-se a decodificação de vídeo, que costuma ter algoritmos de compressão mais complexos em relação aos apresentados em imagens. Ademais, explora-se o fato de que os modelos não utilizam todos os quadros para gerar uma previsão, possibilitando a decodificação de apenas parte do vídeo. Dessa forma, reduziu-se o tempo de época para 40 minutos, com custo de expansão de dados de 7,2 GB para 29,6 GB.

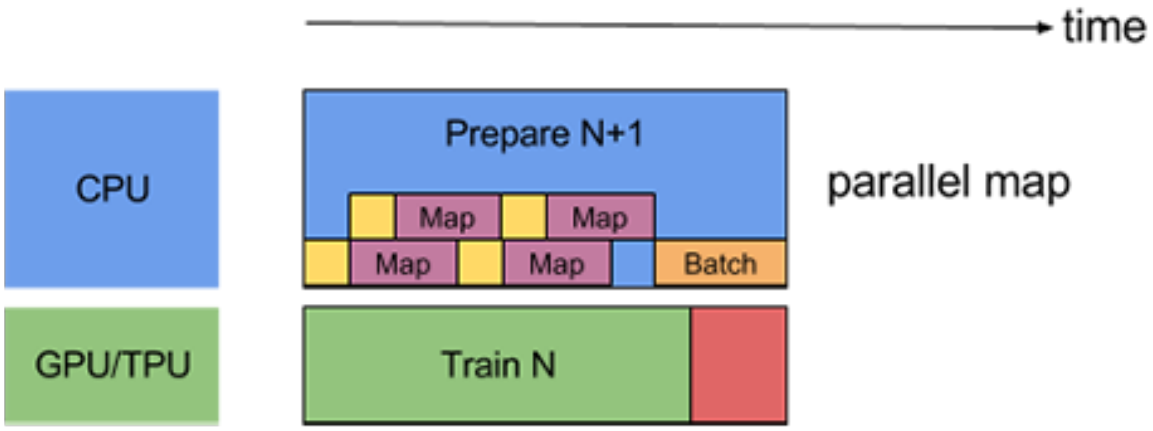


Figura 27 – Implementação otimizada com *pipeline* de processamento paralelo de entrada. Por (ABADI et al., 2015).

Por fim, utilizou-se o formato *TFRecord*, forma de armazenamento binário própria do *Tensorflow* (ABADI et al., 2015), a fim de acelerar a leitura em disco e otimizar o processo de leitura dos dados. Explorou-se o paralelismo a nível da CPU, com leitura dos dados *multicore* e a nível da GPU, processando o próximo conjunto de dados em conjunto com o treino da placa gráfica. O processo descrito é observado na Figura 27.

Desta forma, obteve-se reduções consideráveis, atingindo a marca de menos de 10 minutos por época. Entretanto, há um consumo de espaço em disco consideravelmente maior, chegando aos 282,9 GB.[1]

4.2.3 Limitações de *Hardware*

Outro fator limitante no desenvolvimento do trabalho decorre da memória e poder de processamento disponíveis na placa gráfica utilizada (NVIDIA GeForce GTX 1080 TI). Observa-se em trabalho similares o uso recorrente de múltiplas placas gráficas, com disponibilidade de 16-64 GPUs (CARREIRA; ZISSERMAN, 2017b), 8 GPUs (HARA; KATAOKA; SATOH, 2017), (DIBA et al., 2017) e 4 GPUs (SIMONYAN; ZISSERMAN, 2014a) reportadas.

A limitação encontrada com a utilização de uma única placa gráfica disponível implica em redução do tamanho do *mini-batch* utilizado, redução no número de quadros do vídeo, redução na complexidade do modelo e maior tempo de treino. Dessa forma, tem-se limitações que depreciam significativamente a qualidade das experimentações possíveis.

4.3 Experimentos

Para obtenção de redes capazes de classificar vídeos de acordo com ações realizadas, treinou-se múltiplas arquiteturas apresentadas na seção 3.2. Durante os experimentos, foram exploradas combinações diversas de hiperparâmetros de treinamento, bem como variações nas arquiteturas discutidas, como ponto de extração das características pelas CNNs e quantidade de camadas densas.

4.3.1 Resultados

Esta seção terá como foco a apresentação dos resultados obtidos após a realização dos diversos experimentos com as arquiteturas propostas, observadas inicialmente as metodologias que não incorporam a informação temporal e posteriormente modelos que utilizam tal relação presente nos vídeos.

[1] Neste armazenamento está incluído *data augmentation* nos exemplos de treinamento, procedimento que incorre em maior uso de disco.

4.3.1.1 Quadro Único

Testes iniciais foram realizados com arquiteturas desenvolvidas para classificação de imagens, desconsiderando relações temporais específicas aos vídeos. Como discutido na Seção 3.2.1, propôs-se a experimentação com as variações VGG, InceptionV3 e ResNet50 no *data set* UCF101.

Acurácia (%)	VGG	Inception	ResNet
Treino	95,01	90,16	90,48
Validação	66,34	72,40	69,97
Teste	-	77,11	75,39

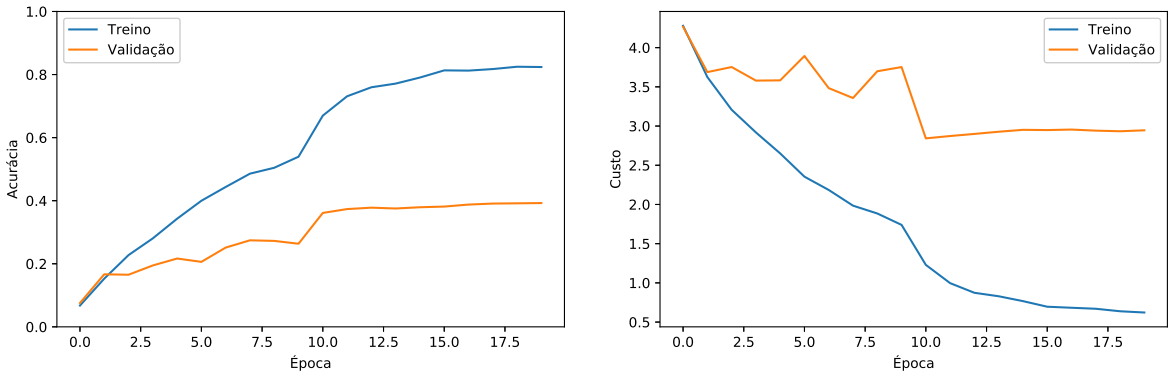
Tabela 1 – Acurácia de modelos de quadro único no conjunto UCF101

Na prática, a precursora VGG apresentou o pior desempenho, fato esperado pela superioridade conhecida das variantes convolucionais mais modernas. Ao final do processo de treinamento, obteve-se $66,34\%$ de acurácia no conjunto de validação do *data set*[2]. Em sequência, a ResNet apresentou acurácia de $69,97\%$ no conjunto de validação do UCF101 quando realizada a classificação utilizando apenas um quadro aleatório. Por fim, a Inception mostrou-se a superior neste conjunto, com $72,40\%$ na validação. O resumo dos resultados é apresentado na Tabela 1.

Apesar de sua simplicidade, observa-se que a média das predições de quadros únicos favorece consideravelmente a acurácia final obtida, apresentada no campo Teste da Tabela 1. Ganhos consideráveis de $4,71$ e $5,42$ pontos percentuais para as arquiteturas Inception e ResNet, respectivamente, são obtidos com esta metodologia, que se baseia apenas em um comportamento análogo a votação de classificação ponderada pela confiança.

A fim de melhor compreender os resultados apresentados, treinou-se a rede com o melhor resultado obtido, Inception, sem utilizar de *transfer learning* do dataset ImageNet, i.e. com seus pesos inicializados aleatoriamente. Foi evidente a influência desta técnica quando observada a acurácia de validação em torno de $39,25\%$ sem a mesma, enquanto obteve-se com a rede pré-treinada acurácia de $72,40\%$, no mesmo conjunto de dados. Ainda, considerando a acurácia no treino, têm-se perda de performance com $82,38$ em detrimento dos $90,16$ pontos percentuais anteriores.

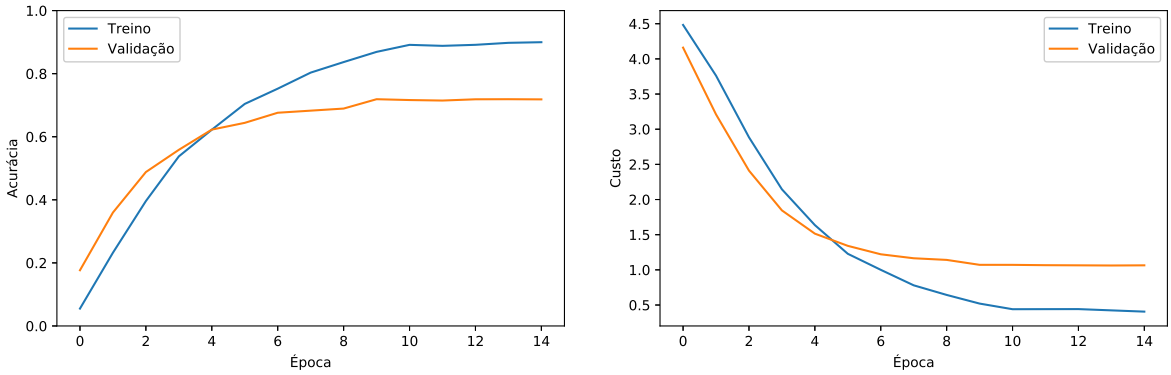
[2] Os testes foram realizados fazendo-se a média da classificação dada pela rede em todos os blocos que compõem o vídeo, nos casos em que não está disponível a acurácia este teste não foi realizado por falta de tempo hábil de implementação.



(a) Acurácia (b) Função Custo

Figura 28 – Curvas de aprendizado da InceptionV3 inicializada aleatoriamente

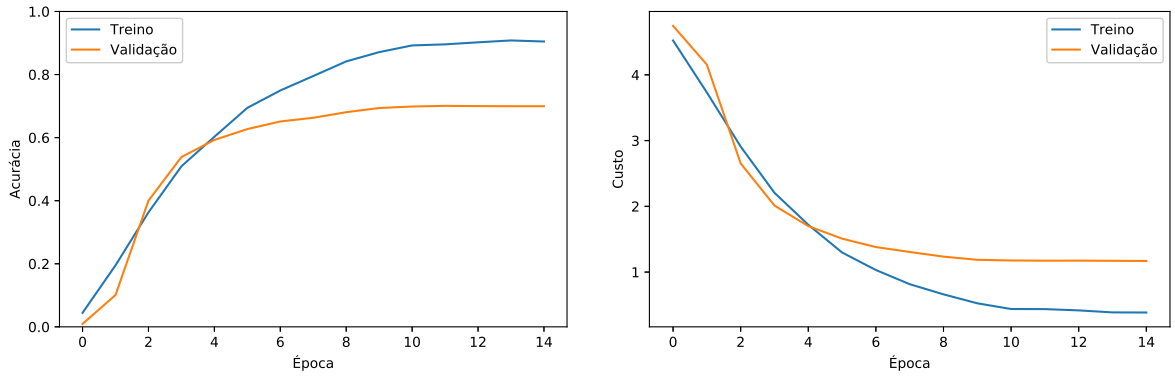
Esta discrepância poderia ser atribuída a necessidade de um maior número de épocas de treinamento, dado que a rede não possui aproveitamento dos pesos e a quantidade de épocas utilizadas é próxima nos dois casos. Entretanto, observa-se na Figura 28 uma estagnação do comportamento do modelo, com saturação das curvas de treino e validação, indício de que não haveria ganho suficiente com mais épocas. Garante-se, pois, a constatação de que grande parte do desempenho deste modelo se dá pelo *transfer learning*.



(a) Acurácia (b) Função Custo

Figura 29 – Curvas de aprendizado da InceptionV3 pré-treinada

Quando se compara o comportamento observado na Figura 28 dado o histórico de treinamento da InceptionV3 com o aproveitamento de pesos por *transfer learning*, mostrado na Figura 29, é evidente um ganho significativo desde o início do processo, tanto na validação quanto no treinamento. Ademais, verifica-se uma significativa melhora da generalização da rede ao observar que a diferença entre os desempenhos de treinamento e validação é bem menor na rede pré-treinada. Vê-se portanto que a transferência de conhecimento auxilia no processo de regularização, evitando o *overfitting*.



(a) Acurácia (b) Função Custo

Figura 30 – Curvas de aprendizado da ResNet50

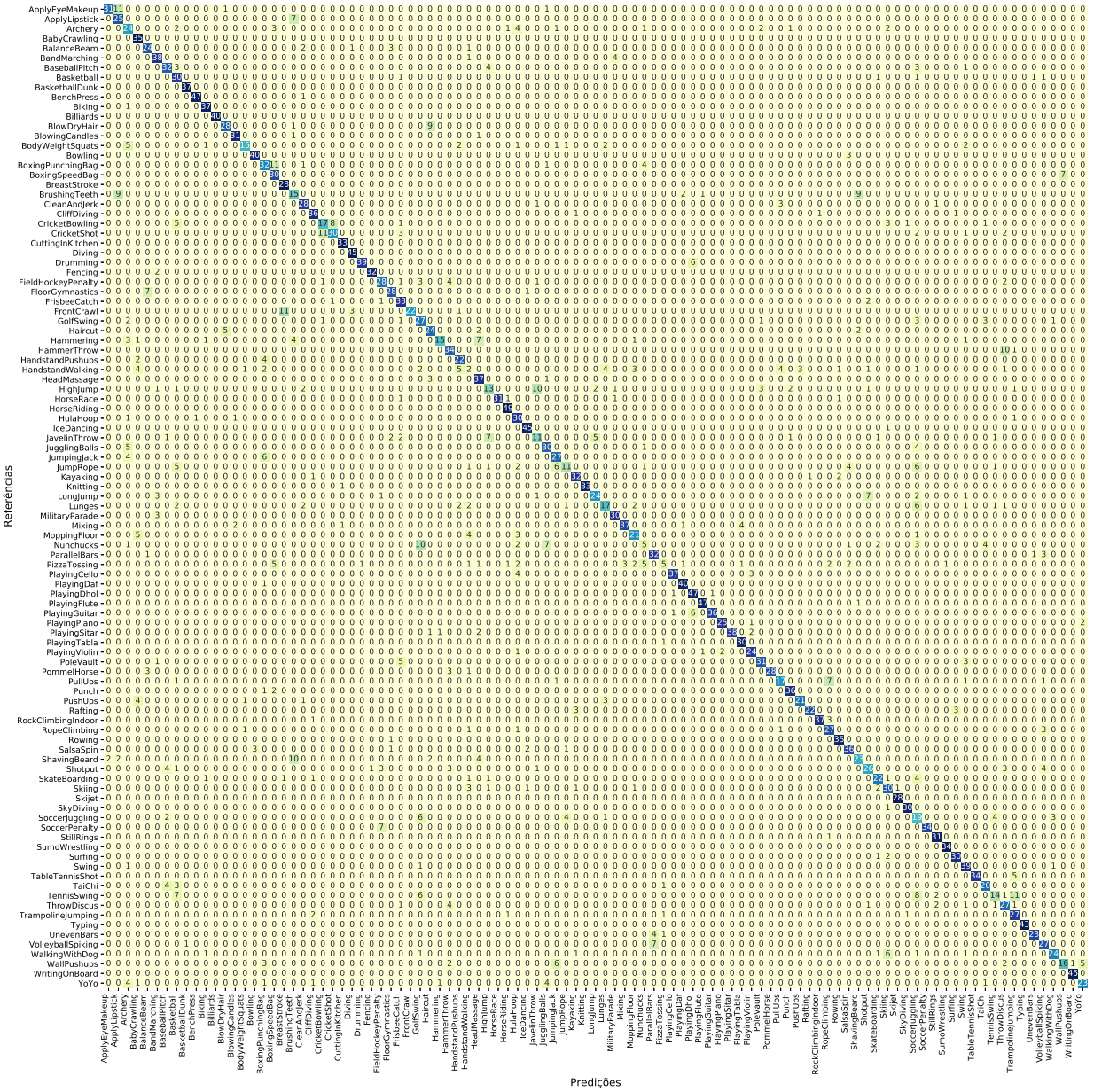


Figura 31 – Matriz de Confusão InceptionV3

Comportamento de treino análogo a Inception é observado na ResNet, vide Figura 30. Apresenta-se também a matriz de confusão[3] do conjunto de testes da Inception, Figura 31, da qual é possível verificar as categorias com menor taxa de acerto, bem como as classes em que os sistema mais se confunde.

4.3.1.2 Múltiplos Quadros

Estabelecidas as capacidades puramente espaciais, procura-se integrar na rede a informação temporal pelo uso de modelos cuja entrada é composta por diversos quadros. Como discutido na Seção 3.2.2, propôs-se a experimentação com as variações CNN-LSTM, I3D e Two-stream no *data set* UCF101.

Acurácia (%)	CNN-LSTM	I3D	Two-stream
Treino	96,91	97,57	99,51
Validação	72,51	88,19	74,47
Teste	-	92,57	-

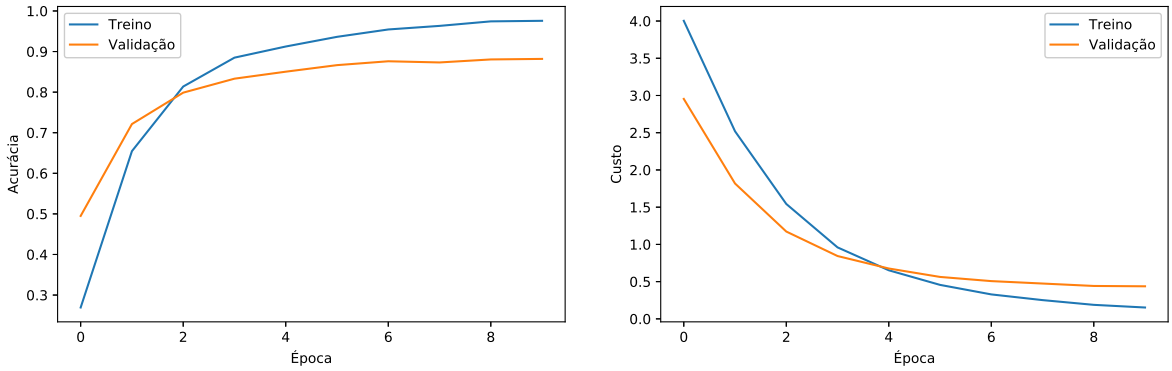
Tabela 2 – Acurácia de modelos de múltiplos quadros no conjunto UCF101

Comparando os desempenhos das redes de múltiplos quadros por meio da Tabela 2 é possível perceber que no conjunto de treinamento o desempenho é consideravelmente próximo, principalmente em oposição ao desempenho de validação da I3D em comparação às outras. Constata-se então uma capacidade de generalização muito maior na I3D, fato que definiu sua posição de melhor rede explorada.

Levando em consideração a maior acurácia obtida com o modelo convolucional Inception na etapa anterior, optou-se por utilizá-lo na fase de extração de características de cada *frame* de entrada da rede LSTM Convolucional. Assim, descartou-se camadas finais deste modelo, utilizando sua representação interna como *features* de entrada para a camada recorrente.

A abordagem mais imediata de treinar diretamente a rede completa, partes convolutiva e recorrente, não apresentou resultados favoráveis. Em todos os testes realizados sua acurácia ficou consideravelmente abaixo do modelo de quadro único do Inception. Em tentativa posterior, treinou-se separadamente a parte convolucional, como apresentado na Seção 4.3.1.1, utilizando tal modelo ajustado para extrair as características. Feito isso, aplica-se a representação em espaço latente do Inception como entrada para a rede recorrente, treinando-a somente a partir desta etapa. Com esta abordagem foi obtido um ganho marginal em relação ao Inception, com $72,51\%$ de taxa de acerto na validação em oposição aos $72,40\%$ observados anteriormente.

[3] Matriz de confusão apresenta a relação entre a classe real e a classe inferida pelo modelo para todos os exemplos disponíveis no conjunto. Explicita casos com maior quantidade de erros e quais pares de classes causaram maior confusão para o modelo, bem como indica falsos positivos e falsos negativos.



(a) Acurácia (b) Função Custo

Figura 32 – Curvas de aprendizado da I3D no UCF101

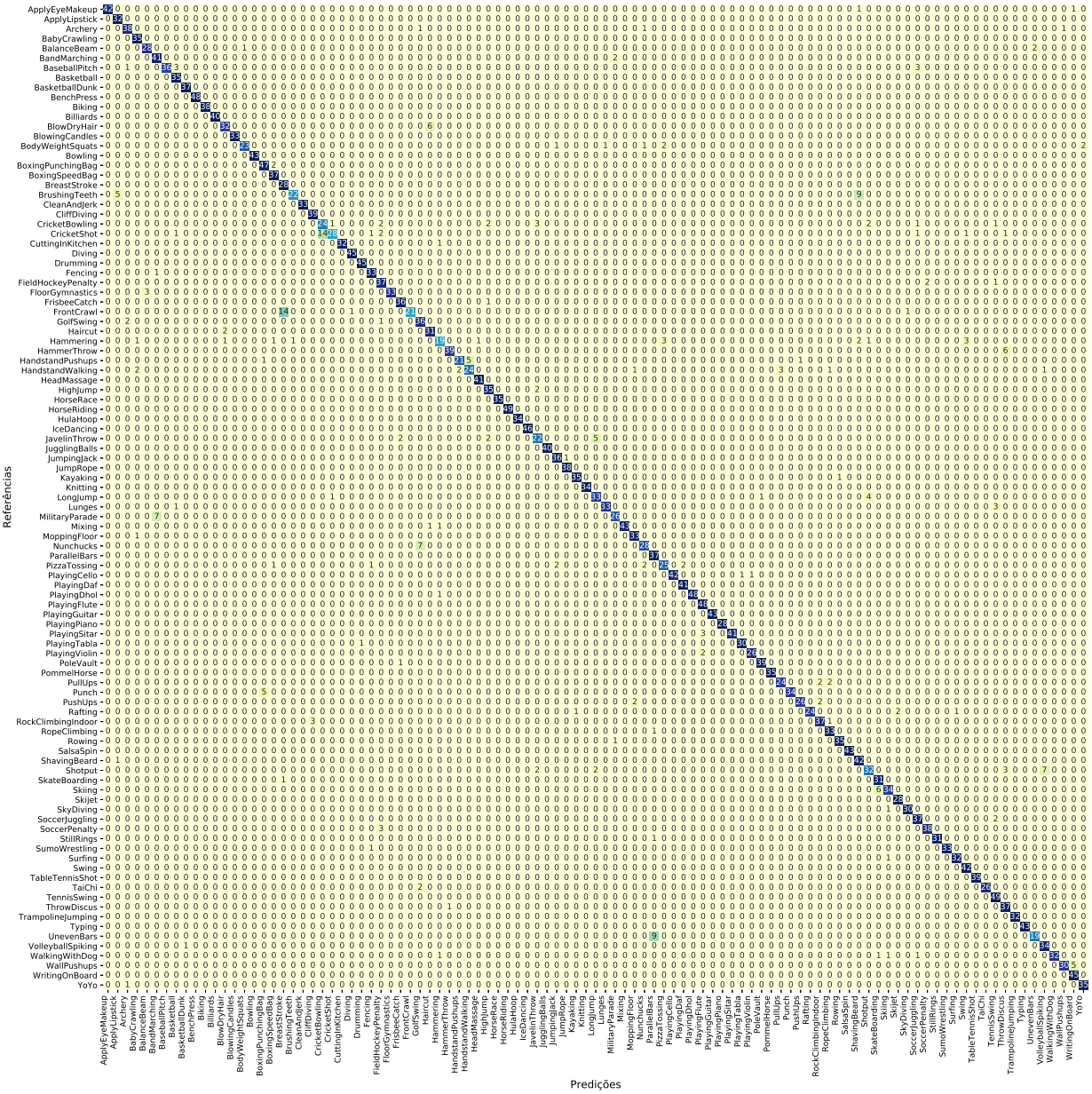


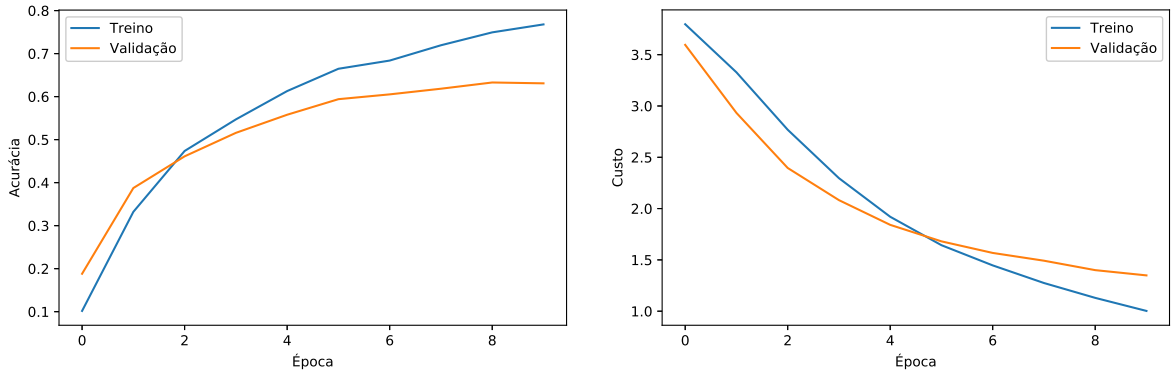
Figura 33 – Matriz de Confusão I3D no UCF101

Em testes posteriores, com o modelo I3D discutido na Seção 3.2.2.2, atingiu-se acurácia de $88,19\%$ no conjunto de validação do UCF101, quando avaliado utilizando apenas um conjunto aleatório de 16 *frames* sequenciais de cada vídeo como entrada ao modelo, a fim de realizar a classificação. A observação das curvas na Figura 32 mostra que este apresenta também melhor performance no conjunto de treino, com $97,57\%$ de acerto. Entretanto, o ganho na curva de treino não é tão expressivo como na validação, já que arquiteturas previamente testadas apresentavam acurácias próximas ou superiores a 90% no conjunto de treinamento.

Desta forma, o ganho mais significativo que se tem é dado na regularização da rede, aproximando os desempenhos nos conjuntos de treinamento e validação, demonstrando uma maior capacidade de generalização. Destaca-se desta forma a importância da utilização de *transfer learning*, quando disponível.

No grupo de testes, divide-se cada vídeo em conjuntos de 16 quadros, de modo que a predição de um filme como um todo é dada pela média das predições dos conjuntos de *frames*. Com tal procedimento para gerar as devidas classificações, obteve-se o expressivo resultado de $92,57\%$ de acurácia, compatível com atual estado da arte (98%) neste *data set* dada a limitação de *hardware* disponível.

A Figura 33 apresenta a matriz de confusão da rede, mostrando que ainda há alguns casos de significativa dificuldade para a mesma, mas que é consideravelmente mais regular que as vistas anteriormente. Destaca-se a concentração de erros na confusão de quatro pares distintos de classes: (FrontCrawl, BreastStroke), (PizzaTossing, UnevenBars), (CricketShot, CricketBowling) e (BrushingTeeth, ShavingBeard), dos quais apenas o segundo não apresenta similaridade ao olho humano.



(a) Acurácia (b) Função Custo

Figura 34 – Curvas de aprendizado da I3D no HMDB51

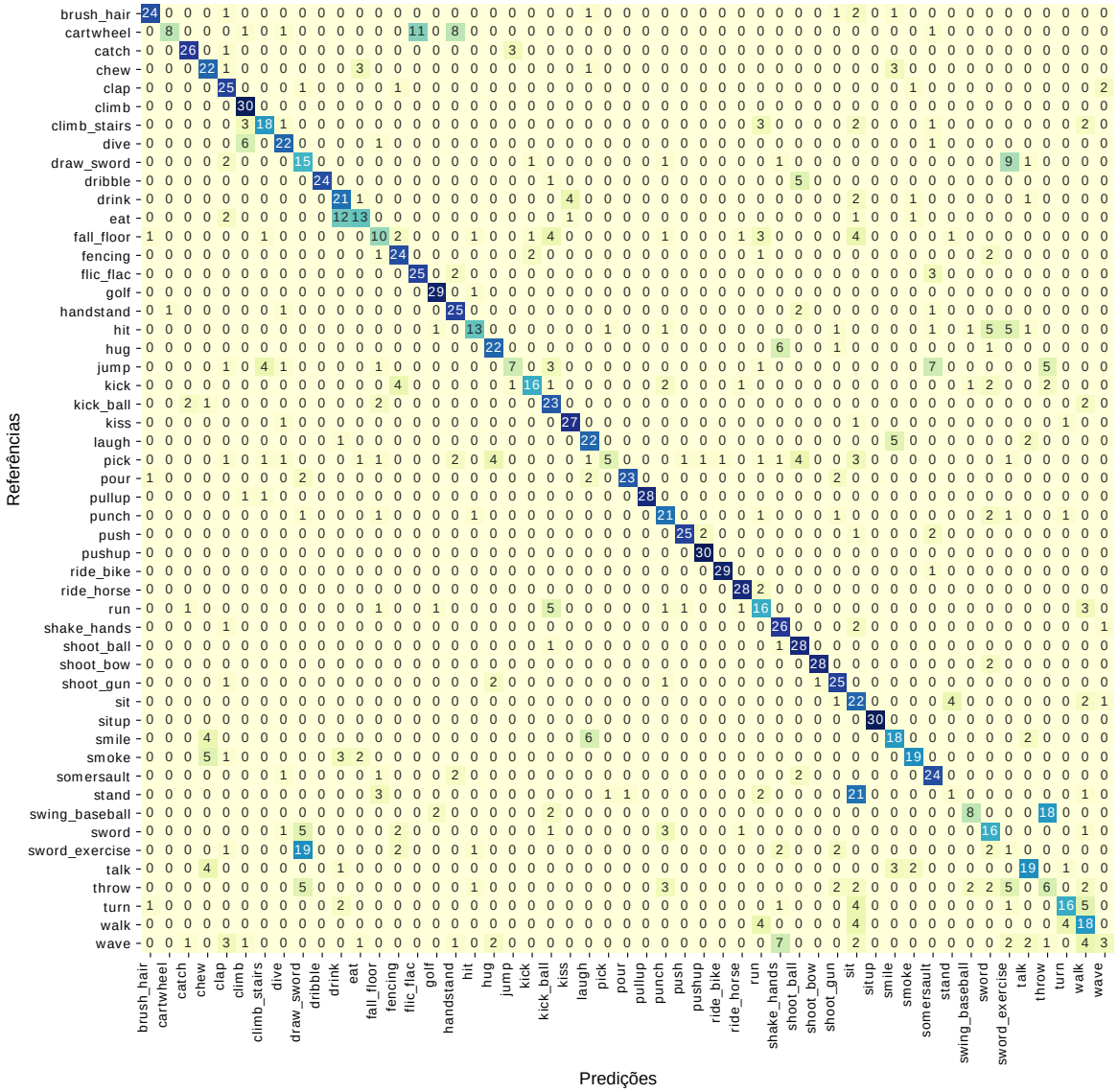


Figura 35 – Matriz de Confusão I3D no HMDB51

A título de curiosidade, experimentou-se a I3D no *data set* HMDB51 para observar o rendimento da mesma em outro conjunto de dados. Esta rede foi selecionada por apresentar um desempenho superior a todas as outras testadas aqui. Neste caso a acurácia obtida foi de $63,09\%$ na validação utilizando um bloco aleatório para classificação.

A Figura 34 mostra a evolução da rede no *data set* HMDB51, evidenciando resultados significativamente inferiores, resultado coerente com o expressado em pesquisas prévias (CARREIRA; ZISSERMAN, 2017b). Utilizando todos os blocos do vídeo é obtida uma acurácia de $65,62\%$, ganho de $2,53$ pontos percentuais, ilustrando a matriz de confusão da rede na Figura 35. Casos de erro evidenciados sugerem que a codificação do movimento não foi devidamente aprendida pela rede, destacando-se os maiores erros nos pares (sword excercise, draw sword) e (sit, stand).

Variação do modelo de Arquitetura com Representação Dupla (SIMONYAN; ZISSERMAN, 2014a), Figura 12, foi a última das abordagens testadas, alterando no entanto a representação do fluxo óptico proposta no artigo, onde se representa as componentes vertical e horizontal dos vetores, enquanto aqui são utilizadas magnitude e fase. Infelizmente os resultados obtidos não foram muito satisfatórios, mantendo-se na faixa de 20 *a* 25% de acurácia. Possíveis influências para este resultado vêm do fato de não ser utilizado *transfer learning* nesta rede e a quantidade de épocas ser consideravelmente baixa por questões de tempo disponível. A alternativa de se utilizar a InceptionV3 para a torre do quadro de referência e manter a sugestão de Simonyan e Zisserman (2014a) para o *optical flow* mostrou resultados superiores, alcançando acurácia de $74,47\%$.

É relevante indicar que, devido a capacidade computacional, não foi possível reproduzir totalmente os métodos de treinamento propostos. Fez-se necessário limitar a quantidade de quadros do *optical flow* a no máximo 8, ocupando a maior parte da memória disponível. Assim, restringiu-se também o uso de valores de *mini-batch* a menos de 16 amostras por atualização, prejudicando o processo de aprendizado. Outro aspecto importante enfrentado fora o tempo de treinamento, chegando a mais de 36 horas no *hardware* utilizado em alguns casos, criando barreiras para múltiplos testes.

Investigações futuras devem ser realizadas a fim de compreender motivo do insucesso da primeira abordagem. Além disso, acredita-se que o treino de redes com apenas um quadro de fluxo ótico de entrada, e posterior fusão das predições como feito na Seção 4.3.1.1 para imagens RGB, deve incorrer em resultados superiores aos aqui apresentados. A utilização do *Tf Records*, implementado posteriormente, também poderia favorecer consideravelmente a eficácia dos testes.

4.3.2 Discussões

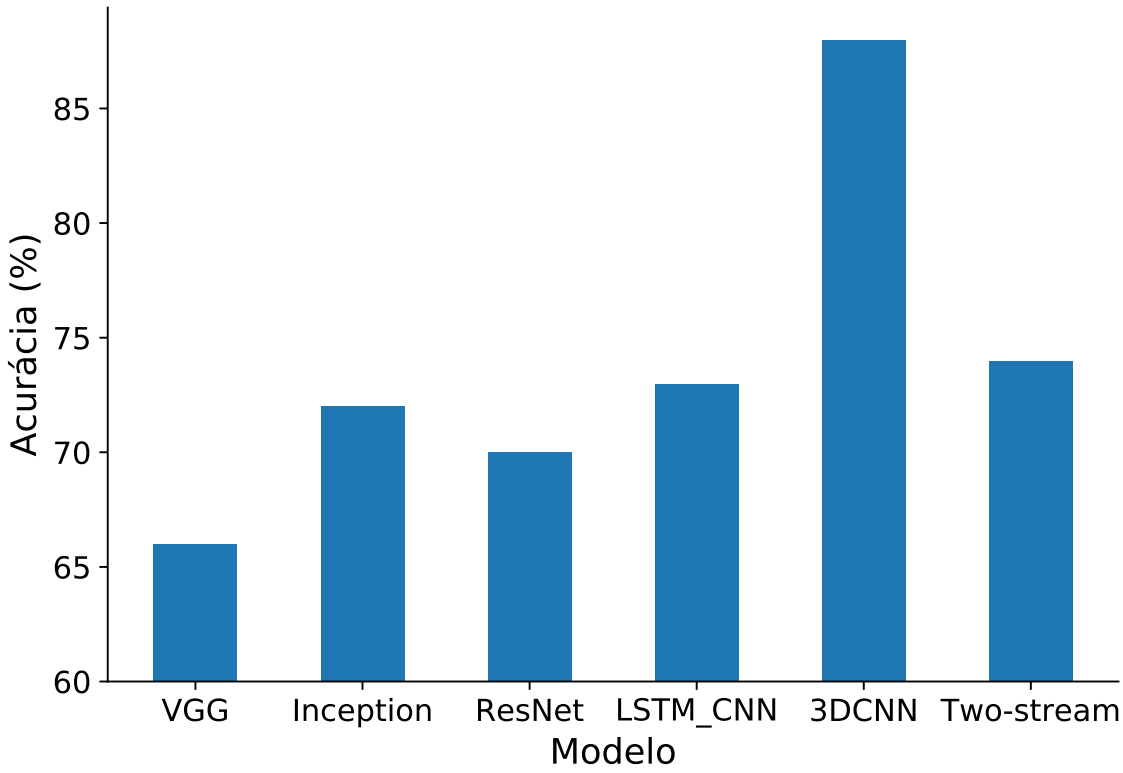
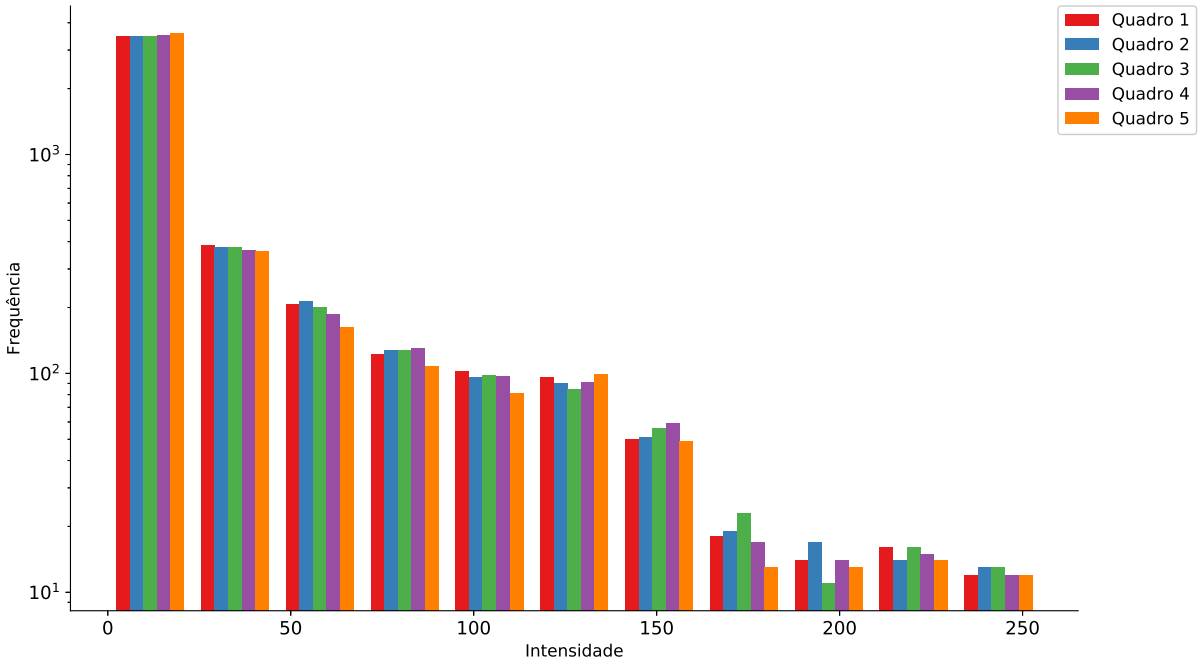


Figura 36 – Resumo de desempenho das redes investigadas

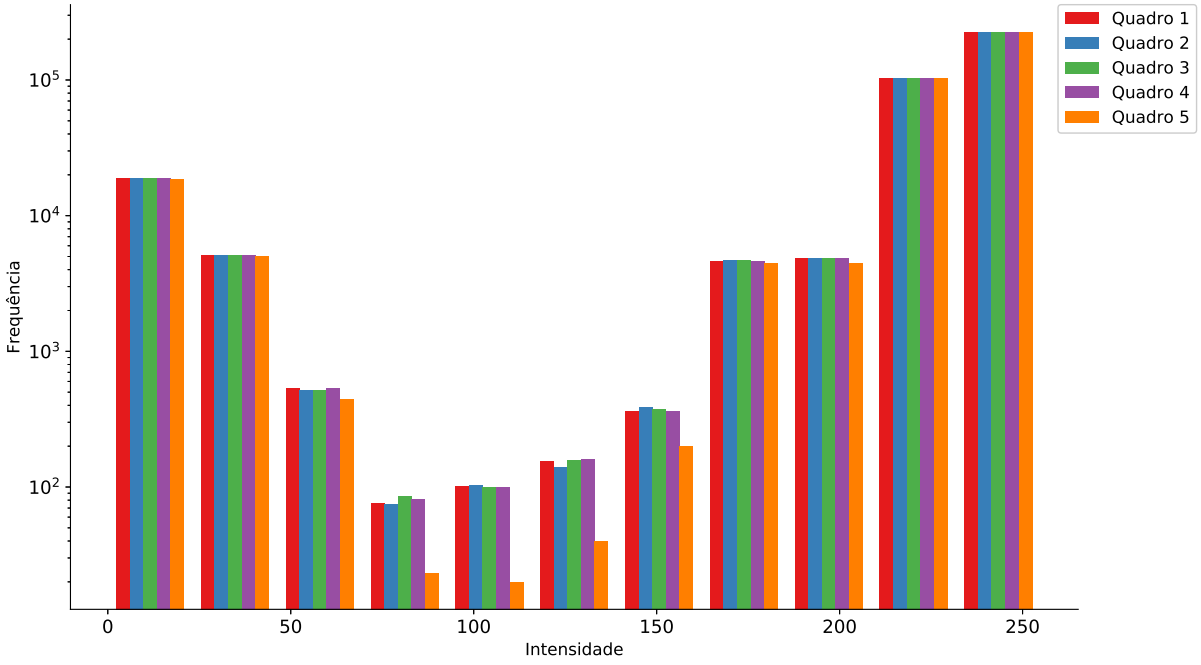
Comparando os diferentes resultados obtidos pelas diversas redes desenvolvidas ao longo do trabalho, ilustrados na Figura 36, pode-se destacar algumas considerações a respeito do uso de inteligência artificial para identificação de ações em vídeo.

Uma característica evidenciada consiste na relevância da informação espacial para determinar o que ocorre em uma cena. Apesar de não apresentar desempenho extraordinário, redes que procuram classificar utilizando apenas um quadro alcançaram acurácia expressiva, com 72% em *data sets* limpos como o UCF101. Tendo em vista que a base de acertos de escolhas aleatórias é próxima a 1%, e descarta-se completamente a informação temporal, há de fato um ganho considerável.

O fato do modelo Convolucional Recorrente apresentado na Seção 3.2.2.1 não ter sido capaz de apresentar melhora significativa em relação a melhor arquitetura de quadro único também é curioso. Visto que redes recorrentes são desenvolvidas com o objetivo de reter informação de instantes anteriores para determinar a classificação da sequência, é razoável esperar um ganho expressivo dada a consideração das relações temporais do vídeo.



(a) Autoencoder



(b) Inception

Figura 37 – Histograma de ativações latentes do *autoencoder* e Inception

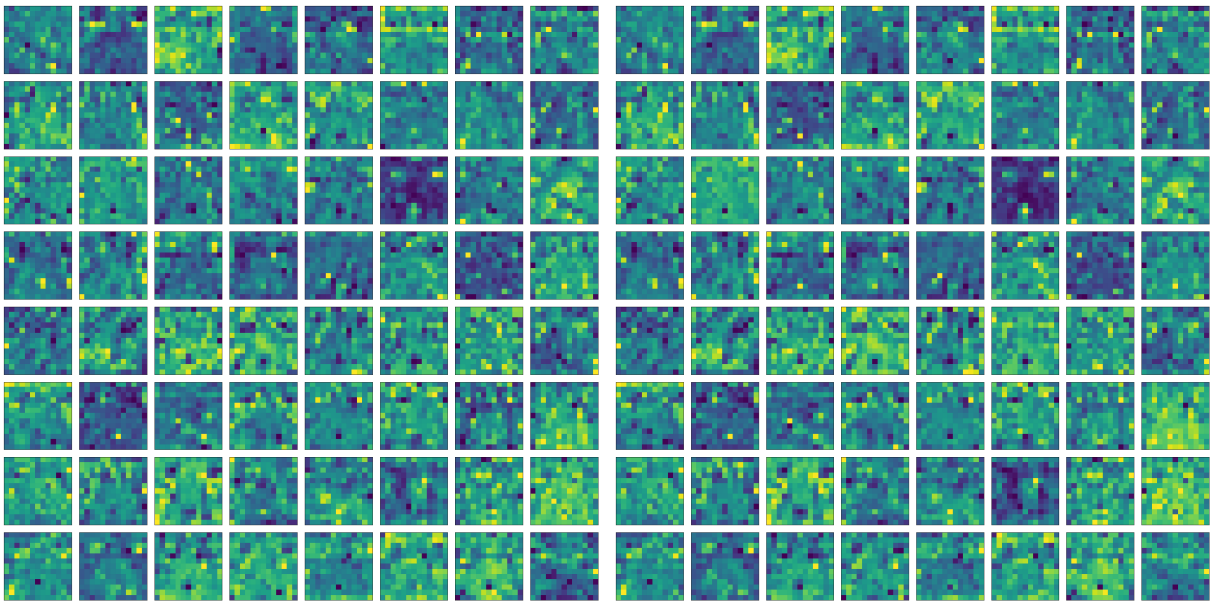
Uma hipótese levantada como possível explicação para este fenômeno é oriunda da etapa de extração de características do espaço latente. Uma das propriedades de redes convolucionais é sua invariância à posição. Esta invariância é um ponto positivo quando o objetivo é a identificação de elementos em um cenário estático, visto que muitas vezes o local do elemento não é relevante para sua caracterização, afinal, um gato é um gato em qualquer local da imagem.

Assim, é razoável imaginar que a rede recorrente não consiga identificar características temporais, visto que nos vídeos a informação temporal é codificada por meio da movimentação, da mudança de posição dos elementos. Uma possível metodologia proposta para contornar este problema consiste na utilização de *autoencoders*, que precisam conter a informação espacial para reconstrução de imagens. A ideia é, ao utilizar o mapeamento latente comprimido de um *autoencoder*, extrair características relevantes e manter a noção espacial das mesmas, de modo que uma relação temporal ao longo dos quadros é estabelecida.

Como *autoencoders* não realizam classificação de elementos, seu treinamento é feito de forma consideravelmente distinta, observando-se a qualidade da imagem após a decodificação em comparação com a original. Isto requer um tratamento diferenciado dos dados e estudo para implementação adequada. Testes preliminares foram realizados a fim de compreender a fundamentação da hipótese, de modo que treinou-se uma rede *autoencoder* convolucional. Analisando o histograma do espaço latente, Figura 37, de um codificador *autoencoder* e uma Inception treinada para classificação de imagens, observa-se que o primeiro apresenta maiores variações, sugerindo mapeamento espacialmente consistente.

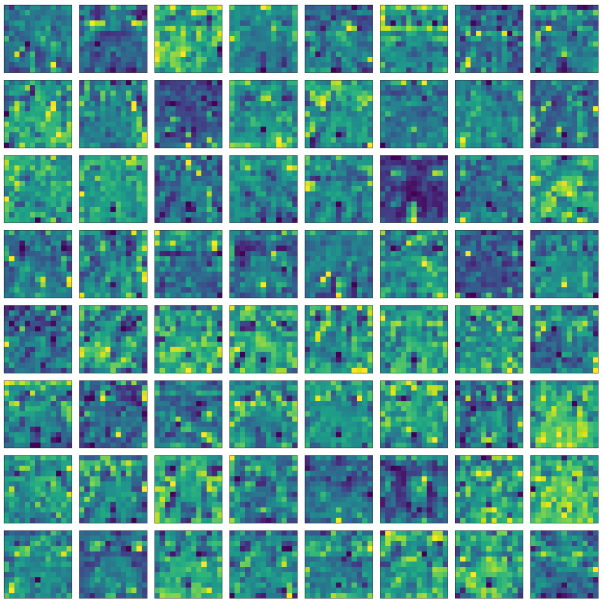
Figura 38 – Quadro exemplo do vídeo original

No intuito de analisar a variação temporal das redes testadas, observando suas ativações intermediárias para diferentes instantes de tempo, aplicou-se imagem apresentada na Figura 38, assim como seus quadros subsequentes, à entrada das redes Inception e I3D. Desta forma, extraiu-se o resultado dos mapas de características internos destes modelos, após aplicações sucessivas de camadas convolucionais.

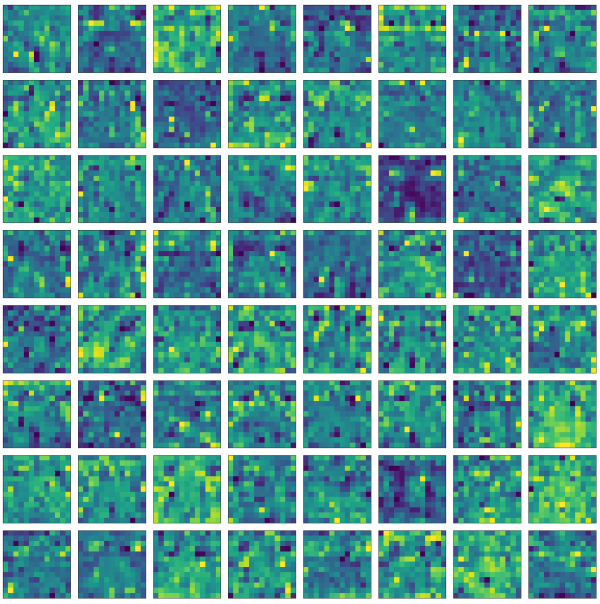


(a) Quadro 1

(b) Quadro 2



(c) Quadro 3



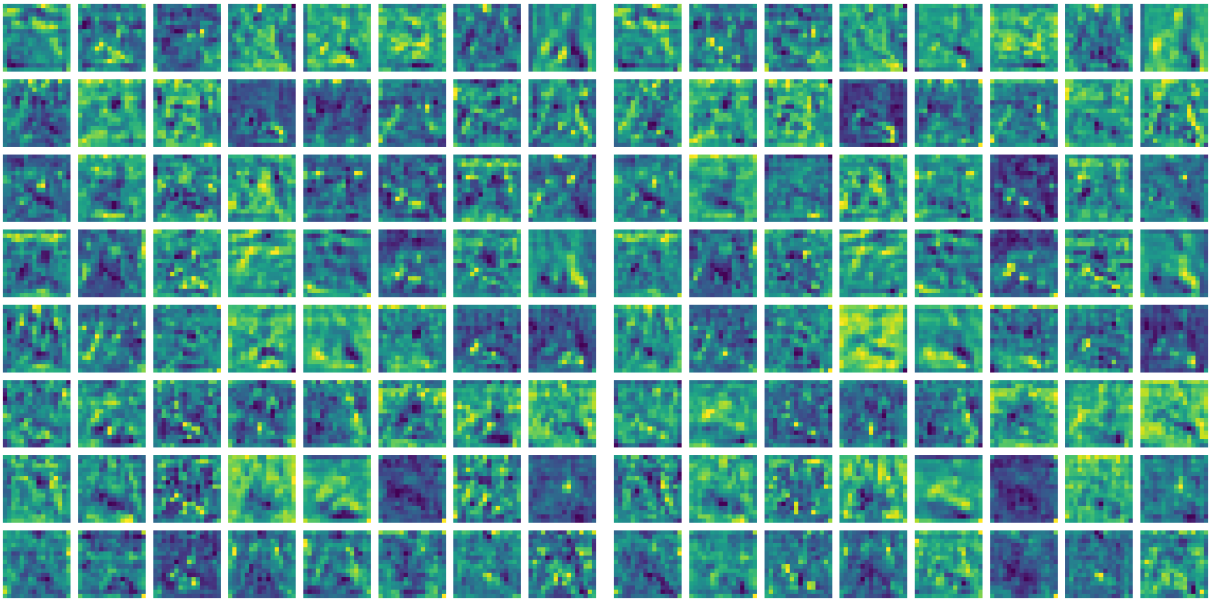
(d) Quadro 4

Figura 39 – Sequência de ativações da Inception

No caso da Inception, em estudo apresentado na Figura 39, aplicou-se 4 *frames* distintos, observando 64 mapas de características intermediários gerados a partir desta entrada. Em cada Figura 39a a 39d, tem-se o resultado da aplicação de um quadro de entrada distinto, de modo que ao compará-las é possível verificar a variação temporal nos mapas de características gerados pela Inception.

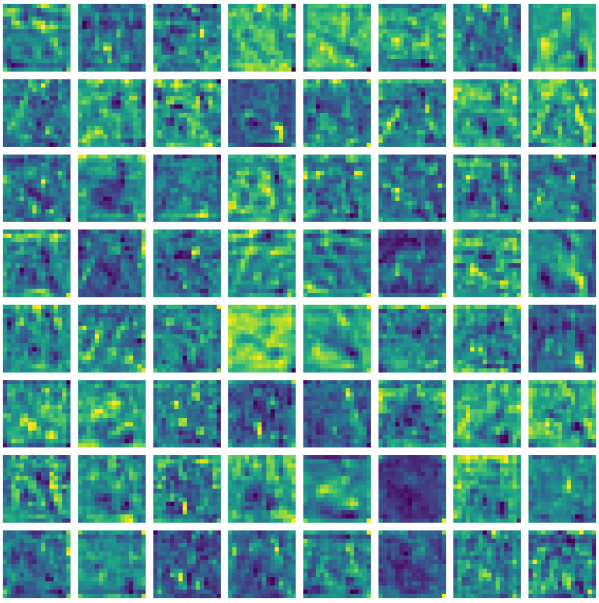
Em observação visual, é fácil notar a alta similaridade entre os mapas gerados, indicando que a invariância destas redes pode ser de fato um problema. Recorda-se do resultado obtido na Figura 37b, em que constatou-se a baixa variação destes mesmos mapas por análise de histograma. Assim, tanto visualmente, como por análise de histograma,

nota-se que estas redes apresentam dificuldades em representar variações temporais.

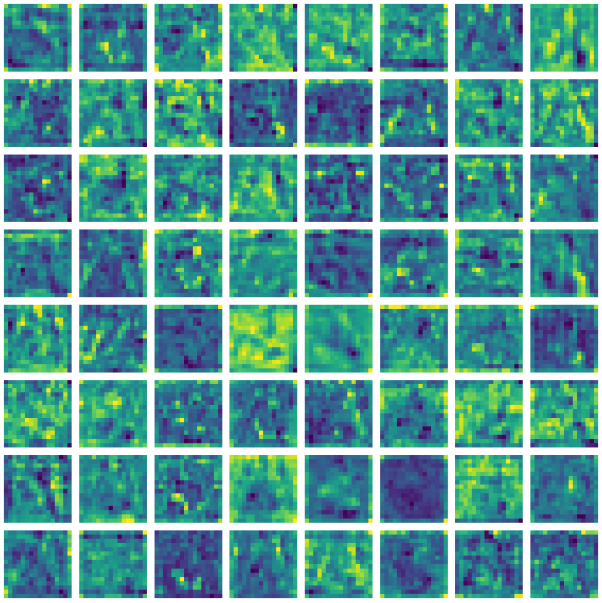


(a) Tempo 1

(b) Tempo 2



(c) Tempo 3



(d) Tempo 4

Figura 40 – Sequência de ativações da I3D

No caso da Figura 40, como a saída possui dimensões (4, 16, 16, 112) as subfiguras correspondem à posição na dimensão temporal da camada e não diretamente a um quadro, de modo que saídas dos filtros foram truncados até o 64º em ambos os casos. Observa-se que as ativações da I3D possuem variação consideravelmente mais notável, de modo que se ressalta a melhor representação temporal destas redes.

Como a I3D realiza convolução temporal, não é possível fazer a correspondência exata dos quadros apresentados a Inception. Entretanto ainda se crê que é válida a ob-

servação da característica temporal desta rede em contraposição à Inception, com ambas atuando próximo ao quadro de referência da Figura 38.

Com a observação das características e desempenhos das diversas redes implementadas, conclui-se que, como esperado, apesar de as características puramente espaciais contribuírem significativamente para a definição de ações, para se ter uma confiabilidade real é preciso tratar também a informação temporal.

Entretanto o tratamento desta informação não é uma tarefa simples, exige certos cuidados e não se pode esperar que a reprodução de tratamentos estáticos ao longo do vídeo capture a relação temporal. Foram exigidas muitas tentativas frustradas até que se chegasse em uma rede adequada, que só foi obtida com a combinação de diversas técnicas de aprendizado.

Arquitetura	UCF101	HMDB51
Two-Stream (SIMONYAN; ZISSERMAN, 2014a)	88,0	59,4
T3D+TSN (DIBA et al., 2017)	93,2	63,5
ResNext-101 (HARA; KATAOKA; SATOH, 2017)	90,7	63,8
ResNext-101 (64f) (HARA; KATAOKA; SATOH, 2017)	94,5	70,2
C3D (TRAN et al., 2015)	90,4	-
Two-Stream I3D (CARREIRA; ZISSERMAN, 2017b)	98,0	80,9
I3D (Nosso)	92,6	65,6

Tabela 3 – Comparação de performance (acurácia %) com modelos estado da arte.

Na Tabela 3 estão listadas as performances de diversas redes de referência vistas no decorrer do trabalho. Uma diferença entre os resultados obtidos é que, excluída a nossa abordagem — que utilizou apenas o *split* 1 —, realizou-se nos outros trabalhos a média nas 3 divisões disponíveis para os *data sets*. Não é esperada mudança considerável nesta divergência, visto que para o caso da Two-Stream I3D (CARREIRA; ZISSERMAN, 2017b) as acurácias obtidas no *split* 1 foram de $98,0\%$ no UCF101 e $81,3\%$ no HMDB51. Todas as redes comparadas nesta tabela se utilizaram de algum tipo de *transfer learning* para obter os resultados representados, ainda que nos respectivos trabalhos se tenha realizado testes sem esta técnica.

É possível notar que os resultados obtidos por meio deste trabalho conseguem ser expressivos mesmo em face a algumas das redes de maior prestígio da área. Em alguns casos, ainda é notória a superação destas redes; embora reitera-se a possibilidade de variações não drásticas caso utilizada a média nos *splits*.

Como caso de maior curiosidade tem-se a ResNext-101 que, com a utilização de 16 quadros, mesma quantidade utilizada na I3D testada neste trabalho, apresenta um desempenho um pouco inferior a esta, ainda que sua variação com 64 quadros a ultrapasse.

Outra peculiaridade é a da T3D+TSN, que apresenta desempenho superior no UCF101 mas inferior no HMDB51.

5 Conclusões e Trabalhos Futuros

Este trabalho explora a implementação de inteligência artificial no contexto de classificação de ações em vídeo. Para este fim foi realizado um estudo a respeito do desenvolvimento histórico deste tipo de aplicação, apresentando a fundamentação necessária para o entendimento do desenvolvimento deste projeto.

Foram investigadas diversas metodologias distintas para atacar o problema em questão e observadas características próprias que cada abordagem apresenta. Parte-se de abordagens simplistas em que a informação temporal é descartada completamente e a classificação é realizada considerando apenas características espaciais de um momento estático. Com esta abordagem é constatado que a informação puramente espacial apresenta uma relevância considerável para a determinação da ação tomada, obtendo resultados expressivos. Entretanto, como esperado, observa-se que esta estimativa ainda apresenta resultados consideravelmente inferiores ao que se deseja.

Prossegue-se então a explorar modelos que levem em consideração, além das características espaciais, a relação temporal presente em vídeos. Algumas arquiteturas que abordam a questão temporal ainda apresentaram desempenhos insatisfatórios, evidenciando a dificuldade do tratamento adequado desta informação. Três metodologias bem distintas foram utilizadas levando em conta o desenvolvimento de ações ao longo dos vídeos, entretanto apenas uma apresentou desempenho realmente significativo. São apresentados alguns questionamentos a respeito de possíveis causas do baixo ganho apresentado em alguns dos modelos espaço-temporais, mas, por limitações de tempo, soluções para estes problemas não puderam ser devidamente exploradas.

Além da implementação destas arquiteturas, discute-se procedimentos disponíveis para auxiliar no treinamento da rede e seu desempenho final. Diversas técnicas que auxiliam na velocidade de treinamento dos modelos foram apresentados, dentre os quais se destacam a forma de tratamento dos dados e o aproveitamento de pesquisas anteriores. Práticas para melhorar o poder de generalização ao final do treinamento também foram utilizadas e mostraram-se bastante influentes para o desempenho obtido, tendo em vista a considerável disparidade presente entre os conjuntos de treinamento e validação apresentada em diversos modelos experimentados, sendo a diminuição desta o que de fato estabeleceu a maior adequação do modelo convolucional 3D.

Com o objetivo de realizar uma pesquisa a respeito da viabilidade e efetividade de se construir e treinar redes de classificação de vídeos, sob contexto de recursos limitados disponíveis a indivíduos independentes, testou-se diversas arquiteturas nos *data sets* UCF101 e HMDB51. Mostrou-se que é possível contornar estas limitações e construir

redes com resultado expressivo, obtendo-se acurácia de $92,6\%$ para o *data set* UCF101, ainda que um pouco abaixo do estado da arte, que apresenta acurácia de $98,0\%$.

Estipulações para trabalhos futuros se concentram em estudos a respeito da utilização de inteligência artificial em treinamentos não supervisionados, compressão de vídeos e possível integração entre a compressão e a classificação de ações em vídeo. Disponibiliza-se também os códigos desenvolvidos (CARVALHO; MARQUES, 2018) de forma a criar acessibilidade ao mesmo e facilitar novos trabalhos complementares na área de classificação de ações em vídeo e inteligência artificial em problemas gerais.

Referências

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado 5 vezes nas páginas 9, 18, 49, 50 e 51.

ALPAYDIN, E. *Introduction to Machine Learning*. 3. ed. Cambridge, MA: MIT Press, 2014. (Adaptive Computation and Machine Learning). ISBN 978-0-262-02818-9. Citado 2 vezes nas páginas 24 e 26.

BANERJEE, S. An introduction to recurrent neural networks. *Medium*, 2018. Citado 2 vezes nas páginas 9 e 31.

CARREIRA, J.; ZISSERMAN, A. *I3D models trained on Kinetics*. 2017. <https://github.com/deepmind/kinetics-i3d>. Citado na página 49.

CARREIRA, J.; ZISSERMAN, A. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017. Disponível em: <http://arxiv.org/abs/1705.07750>. Citado 7 vezes nas páginas 9, 37, 42, 44, 51, 58 e 65.

CARVALHO, G.; MARQUES, P. *Deep Video*. 2018. <https://gitlab.com/gustavohtac/video-classification>. Citado 2 vezes nas páginas 49 e 68.

CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>. Citado 2 vezes nas páginas 18 e 49.

CHRISTIAN, S. et al. Rethinking the inception architecture for computer vision. *CVF*, 2015. Citado na página 40.

CISCO. *White paper: Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. [S.l.], 2018. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>. Citado 3 vezes nas páginas 9, 17 e 18.

DENG W. DONG, R. S. L.-J. L. K. L. J.; FEIFEI, L. Imagenet: A large-scale hierarchical image database. In: *CVPR*. [S.l.: s.n.], 2009. Citado 2 vezes nas páginas 37 e 38.

DIBA, A. et al. Temporal 3d convnets: New architecture and transfer learning for video classification. *CoRR*, 2017. Citado 3 vezes nas páginas 37, 51 e 65.

DONAHUE, J. et al. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014. Disponível em: <http://arxiv.org/abs/1411.4389>. Citado 2 vezes nas páginas 9 e 42.

ESE. *keras-kinetics-i3d*. [S.l.]: GitHub, 2018. <https://github.com/dlpbc/keras-kinetics-i3d>. Citado na página 49.

HARA, K.; KATAOKA, H.; SATOH, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *CoRR*, abs/1711.09577, 2017. Disponível em: <http://arxiv.org/abs/1711.09577>. Citado 5 vezes nas páginas 9, 42, 43, 51 e 65.

HE, K. et al. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado 4 vezes nas páginas 9, 38, 40 e 41.

HENKE, N.; LIBARIKIAN, A.; WISEMAN, B. *Straight talk about big data*. 2016. Disponível em: <https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/straight-talk-about-big-data>. Citado na página 18.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>. Citado na página 36.

JUNG, A. *imgaug*. [S.l.]: GitHub, 2018. <https://github.com/aleju/imgaug>. Citado na página 32.

KARPATHY, A. *CS231n Convolutional Neural Networks for Visual Recognition*. 2018. Disponível em: <http://cs231n.github.io/convolutional-networks/>. Citado 5 vezes nas páginas 9, 21, 23, 28 e 30.

KARPATHY, A. et al. Large-scale video classification with convolutional neural networks. In: *CVPR*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 9 e 35.

KAY, W. et al. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. Disponível em: <http://arxiv.org/abs/1705.06950>. Citado na página 44.

KENSTLER, B. *Cyclical Learning Rate (CLR)*. [S.l.]: GitHub, 2016. <https://github.com/bckenstler/CLR>. Citado 2 vezes nas páginas 9 e 34.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 25.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Citado 4 vezes nas páginas 17, 23, 36 e 38.

KUEHNE, H. et al. HMDB: a large video database for human motion recognition. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2011. Citado na página 48.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved. SN -, v. 521, p. 436 EP –, May 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Citado 4 vezes nas páginas 9, 17, 22 e 29.

LIN, M.; CHEN, Q.; YAN, S. Network in network. *CoRR*, abs/1312.4400, 2013. Disponível em: <http://arxiv.org/abs/1312.4400>. Citado na página 39.

MANDAL, M. K. Implementing pca, feedforward and convolutional autoencoders and using it for image reconstruction, retrieval compression. *Medium*, 2018. Citado 2 vezes nas páginas 9 e 30.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. 1943. *Bulletin of mathematical biology*, v. 52 1-2, p. 99–115; discussion 73–97, 1990. Citado 2 vezes nas páginas 21 e 22.

NG, J. Y.-H. et al. Beyond short snippets: Deep networks for video classification. In: *CVPR*. [S.l.]: IEEE Computer Society, 2015. Citado 3 vezes nas páginas 9, 36 e 42.

OLAH, C. Understanding convolutions. *GITHUB blog, posted on July*, 2014. Disponível em: <http://colah.github.io/posts/2014-07-Understanding-Convolutions/>. Citado na página 28.

OLAH, C.; MORDVINTSEV, A.; SCHUBERT, L. Feature visualization. *Distill*, 2017. Https://distill.pub/2017/feature-visualization. Citado na página 29.

RADEK, B. Multi-label image classification with inception net. *Medium*, 2017. Citado 2 vezes nas páginas 9 e 40.

ROBBINS, H.; MONRO, S. A stochastic approximation method. *Ann. Math. Statist.*, The Institute of Mathematical Statistics, v. 22, n. 3, p. 400–407, 09 1951. Disponível em: <https://doi.org/10.1214/aoms/1177729586>. Citado na página 25.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 22.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group SN -, v. 323, p. 533 EP –, Oct 1986. Disponível em: <https://doi.org/10.1038/323533a0>. Citado na página 22.

SHARMA, A. *What is the differences between artificial neural network (computer science) and biological neural network?* 2017. Disponível em: <https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network>. Citado 2 vezes nas páginas 9 e 21.

SILVA, V. de O. Human action recognition in image sequences based on a two-stream convolutional neural network classifier. In: . [s.n.], 2017. Disponível em: <http://repositorio.unb.br/handle/10482/25201>. Citado na página 36.

SIMONYAN, K.; ZISSERMAN, A. Two-stream convolutional networks for action recognition in videos. In: *NIPS*. [S.l.: s.n.], 2014. Citado 8 vezes nas páginas 9, 35, 36, 45, 49, 51, 59 e 65.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. Disponível em: <http://arxiv.org/abs/1409.1556>. Citado na página 38.

SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. Disponível em: <http://arxiv.org/abs/1803.09820>. Citado na página 33.

SOOMRO, K.; ZAMIR, A. R.; SHAH, M. Ucf101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, 2012. Citado 4 vezes nas páginas 9, 18, 47 e 48.

SZEGEDY, C. et al. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. Disponível em: <http://arxiv.org/abs/1409.4842>. Citado 4 vezes nas páginas 9, 36, 38 e 39.

TRAN, D. et al. Learning spatiotemporal features with 3d convolutional networks. In: *ICCV*. [S.l.]: IEEE Computer Society, 2015. Citado 5 vezes nas páginas 9, 37, 42, 43 e 65.

VO, A. Deep learning – computer vision and convolutional neural networks. *Word Press*, 2018. Disponível em: <https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/>. Citado 2 vezes nas páginas 9 e 28.

WEINBERGER, K. *Lecture notes in CS4780/CS5780 Machine Learning*. [S.l.]: Cornell University, 2015. Citado 2 vezes nas páginas 9 e 27.

ZHONG, Z. et al. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017. Disponível em: <http://arxiv.org/abs/1708.04896>. Citado na página 32.

ZHU, Y. et al. Hidden two-stream convolutional networks for action recognition. *CoRR*, 2017. Citado na página 36.