Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Treinamento e Síntese de FCNs com Campo Receptivo Variável

Um estudo aplicado a segmentação de imagens

Pedro Sola Pimentel

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Roberto Hirata

São Paulo
5 de Dezembro de 2019

Resumo

Pedro Sola Pimentel. **Treinamento e Síntese de FCNs com Campo Receptivo Variável: *Um estudo aplicado a segmentação de imagens***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2019.

Neste trabalho exploramos o uso de Redes Neurais Totalmente Convolucionais para segmentação de imagens combinando diferentes campos receptivos de uma forma unificada. Criamos uma arquitetura de redes neurais que combina estes campos receptivos em um único modelo de segmentação. Nos testes realizados, este modelo obteve performance superior aos modelos individuais de segmentação. Além disso, conseguimos obter resultados melhores que o modelo com área de campo receptivo equivalente em uma parte dos exemplos considerados. Afim de prover uma leitura concisa e suficiente, apresentamos todos os conceitos básicos utilizados durante o projeto, assim como detalhes de assuntos que cerceiam o trabalho e podem ser de interesse do leitor.

Palavras-chave: Segmentação. Aprendizado-de-Maquina. Deep-Learning. Redes-Neurais. FCNs. Convoluções.

Abstract

Pedro Sola Pimentel. **Training and Synthesis of FCNs with Variable Receptive Field: *A study applied to image segmentation***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2019.

Throughout this work we explore the use of Fully Convolutional Networks for image segmentation combining different receptive fields in a unified manner. We've created a Neural Network architecture that combines these receptive fields in one single segmentation model. On the tests, this model achieved superior performance than the individual segmentation models. Also, it was able to achieve better results than the model with equivalent receptive field on part of the considered examples. Aiming to provide a concise and self-sufficient read, we present all the basic concepts used on the project, as well as details on topics that talk to this project and may be of interest to the reader.

Keywords: Segmentation. Machine-Learning. Deep-Learning. Neural-Networks. FCNs. Convolution.

Lista de Figuras

Sumário

Apêndices

Anexos

Capítulo 1

Introdução

A recente disponibilização de recursos computacionais através de plataformas online, denominadas *nuvem*, juntamente com a diminuição de preços e aumento de poder de processamento possibilitou grandes avanços em tópicos relacionados a Inteligência Artificial. Tarefas que antes seriam feitas por profissionais com conhecimento específico da área hoje são realizadas por algoritmos de aprendizado, o que possibilita que estes mesmos profissionais foquem seus esforços em outras áreas do conhecimento e dediquem seu tempo às tarefas mais importantes da profissão.

Em sua maioria, algoritmos de aprendizado dependem de uma gama de exemplos para performar a tarefa desejada. Neste sentido, parte da automatização das tarefas relacionadas à imagens se deve à crescente disponibilização de conjunto de dados deste tipo. Cada imagem funciona como uma fonte de informação para o algoritmo, que emprega transformações como remoção de ruído e destaque de contorno a fim de realçar objetos de interesse. Geralmente essas transformações atuam localmente, em regiões centradas em um pixel e delimitadas por uma janela.

Dentre estas tarefas, podemos descrever desde as mais comuns às mais complexas, variando de segmentação de dígitos escritos à mão (Lu e Shridhar, 1996) até detecção de tumores em exames (Mustaqeem *et al.*, 2012). Para isso, os mais diversos algoritmos são empregados, tais como: regressão linear, clustering, redes neurais, redes neurais convolucionais e SVM. Aqui vamos explorar o problema de segmentação de imagens, que de forma resumida significa classificar pixels de uma imagem, de forma a identificar entidades que possuem alguma característica em comum.

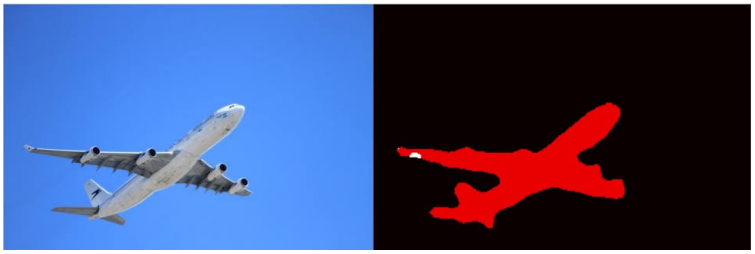

Figura 1.1: *Segmentação de uma imagem de avião.*

Para isso utilizamos redes convolucionais para treinar diferentes classificadores que utilizam janelas com centro ligeiramente deslocado. Esta ideia veio da seguinte observação: Por um lado é matematicamente claro que quanto maior o campo de captura de um classificador de pixels, melhor sua capacidade de discriminação, visto que temos mais informação para realizar a predição. Na prática, contudo, o tamanho de um campo receptivo é limitado em função da quantidade de dados de treinamento (quanto maior o campo receptivo, menor a quantidade de amostras) e a complexidade de rede, visto que a complexidade do classificador cresce conforme aumentamos o campo receptivo, precisando assim de mais exemplos para se obter um classificador robusto. Este fato acaba criando um trade-off entre erro de generalização e erro de completude de amostra.

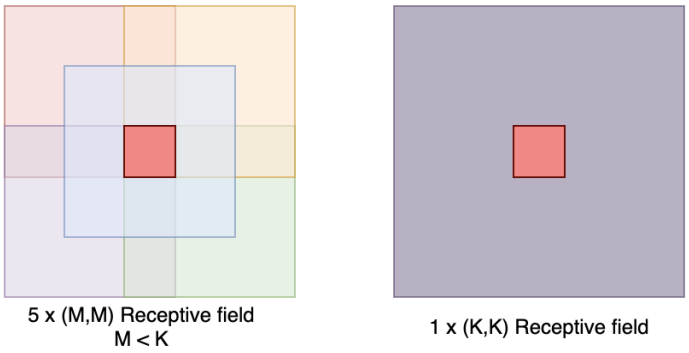

Figura 1.2: *Ideia de campo receptivo variável: À esquerda 5 campos receptivos que juntos totalizam a área captada pelo campo receptivo da direita.*

Quando combinamos vários classificadores que são baseados em janelas com os centros ligeiramente deslocadas em relação ao pixel-alvo, podemos pensar que a combinação está tendo acesso a um conjunto maior de pixels em torno do pixel-alvo, embora cada um dos classificadores tenha acesso a apenas um subconjunto deles. **A Figura 1.1** ilustra esse fato. Em seu artigo Hirata, 2008 utiliza classificadores morfológicos para realizar este mesmo procedimento. Aqui queremos estudar o uso de redes convolucionais para realizar a tarefa de segmentação. Com base nos experimentos já realizados na literatura esperamos que a combinação destes classificadores resulte em um classificador que tenha desempenho ao menos igual ao melhor desempenho dentre os classificadores individuais que fazem parte da combinação.

Neste trabalho aplicamos segmentação de imagens em um conjunto de imagens de *mangás* (revistas no estilo de quadrinhos japonês). O objetivo é extrair os textos de cada uma das páginas disponíveis de forma a reduzir o erro total em relação a imagem ideal. Mais detalhes sobre o conjunto de imagens, tamanho de amostra e detalhes de implementação serão discutidos no capítulo **5**.

Ao decorrer do texto daremos detalhes sobre os conceitos empregados neste processo, arquitetura utilizada e *background* matemático mínimo para entendimento do trabalho. Caso o leitor tenha interesse em algum dos assuntos que cerceiam o conteúdo disponibilizado aqui, é recomendado a leitura do conteúdo citado na seção de bibliografia, assim como revistas acadêmicas relacionadas.

Este texto está organizado da seguinte maneira: O capítulo 2 dá uma visão mais ampla sobre a tarefa que iremos explorar e os conceitos básicos de aprendizado de máquina, enquanto que o Capítulo 3 introduz os conceitos técnicos e matemáticos específicos aos algoritmos que utilizamos no trabalho. Em seguida tocamos na metodologia do trabalho em si, como detalhes de arquitetura e fluxo de informações. Feito isso, apresentamos os resultados obtidos através da técnica escolhida no conjunto de dados especificado.

Capítulo 2

Conceitos Preliminares

2.1 Segmentação de Imagens

Imagens digitais são uma coleção de pixels. Estes elementos são em geral arranjados em uma grade discreta retangular $G \subseteq Z^2$ e possuem, cada um, uma coordenada de linha i, uma de coluna j e valores de intensidade de cor associados. Desta forma, podemos descrever uma imagem matematicamente como uma função $f(G)$:

$$f : Z^2 \rightarrow K^n$$

onde K é o conjunto das intensidades e n o número de bandas. Tipicamente uma imagem é dividida nas bandas **vermelho**, **verde** e **azul** sendo assim denominadas **RGB**. Já as intensidades de cor geralmente variam no intervalo $[0, 255]$.

Além de intensidades, pixels possuem uma relação de vizinhança com outros pixels. Comumente consideram-se as vizinhanças 4 ou 8. Os 4-vizinhos baseiam-se na conectividade 4, que estabelece que a conectividade é definida por adjacência vertical ou horizontal. Já os 8-vizinhos são definidos pela conectividade 8, estabelecida como a adjacência não só na horizontal e vertical, mas também nas diagonais. O conjunto de pixels vizinhos de um pixel p será denotado $V(p)$, e depende da conectividade considerada. Dizemos que um pixel p é conectado a outro $q \iff q \in V(p)$. Um conjunto de pixels conectados, isto é, tal que existe um caminho conectando quaisquer dois pixels do conjunto, é uma **região** ou componente de uma imagem.

O termo segmentação é descrito como o ato de segmentar, ou fracionar algum conjunto ou população em partes que compartilham uma mesma característica. Áreas como *Segmentação de Mercado* e *Segmentação de Marketing* são alguns exemplos onde a técnica de segmentação pode ser empregada, nestes casos o interesse em questão é geralmente criar planos ou campanhas específicas para cada um dos segmentos criados, trazendo um desempenho melhor devido à especificidade.

A segmentação de imagens consiste em particionar um conjunto de pixels em regiões não vazias, duas a duas disjuntas, de tal forma que a união destas regiões cubra todos os pixels da imagem. Em geral a granularidade, ou especificidade, da segmentação é estabele-

cida baseada na aplicação em questão. Cada uma dessas regiões apresenta uma mesma característica, de interesse do pesquisador, como cor, textura ou objeto descrito. Em algumas aplicações o interesse consiste em obter regiões maximais com cores homogêneas, sem necessariamente uma interpretação semântica associada às regiões. Em outras situações, espera-se que as regiões correspondam a objetos ou partes de objetos que constituem o alvo de um estudo. Por exemplo, se o estudo refere-se à análise de imagens de células, pode ser do interesse simplesmente separar células do restante, mas também pode ser do interesse segmentar células e seus núcleos simultaneamente.

2.2 Conceitos de Aprendizado de Máquina

Existem diferentes algoritmos para segmentação de imagens. No entanto, em essência, o propósito de todos eles é o mesmo: o de agrupar ou classificar pixels de forma que o rótulo de classificação dos pixels seja suficiente para identificar as regiões da segmentação. Essa tarefa de classificação de pixels pode ser realizada por algoritmos de aprendizado de máquina. Em uma abordagem mais tradicional, tipicamente extraem-se características diversas da imagem, em geral em torno de cada pixel, e essas características são utilizadas como uma representação do pixel para efeitos de classificação. As características podem ser informações relacionadas às intensidades de cor, textura, localização ou outra característica relativa ao pixel ou conjunto de pixels.

O termo *Aprendizado de Máquina* foi introduzido inicialmente por Solomonoff, 1957. No artigo *An Inductive Inference Machine* o autor descreve Aprendizado de Máquina da seguinte maneira: "*Uma máquina desenhada para aprender a resolver problemas matemáticos através de uma série de exemplos resolvidos corretamente.*". Contudo devido aos avanços recentes na área podemos redefinir Aprendizado de máquina, sem perda de generalidade, como uma aplicação de Inteligência Artificial (IA) que provê a sistemas a habilidade de aprender e melhorar através da experiência, sem ser explicitamente programados.

Ainda que complicado à primeira vista, este pode ser abstraído como uma tarefa simples de encontrar padrões nos dados. Assim como o raciocínio básico, estes algoritmos levam em consideração as particularidades de objetos, conceitos e definições e podem ou não assumir distribuições nos dados. No caso de uma regressão linear por exemplo, assumimos que a distribuição dos dados é linear, e a partir daí criamos maneiras rápidas de calcular a reta que melhor descreve os dados segundo alguma métrica definida.

Tradicionalmente dividimos os algoritmos de aprendizado como **aprendizado supervisionado**, **aprendizado não-supervisionados**, e **aprendizado por reforço**.

2.2.1 Aprendizado Supervisionado

Denominamos como problemas de aprendizado supervisionado aplicações em que o conjunto de treino consiste de um conjunto de vetores de entrada e um conjunto de vetores de saída correspondente. Problemas onde o vetor de saída consiste de uma ou mais variáveis contínuas são denominados problemas de **regressão**. Um exemplo desta categoria é a previsão de demanda de um produto de acordo com uma série de variáveis temporais, como mês, temperatura e presença de campanhas de marketing.

No caso de segmentação de imagens, a forma mais simples de prover esses vetores é criar pares de imagem consistindo de uma imagem de entrada (um exemplar do tipo de imagens que se deseja segmentar) e a correspondente imagem esperada após a segmentação. Essa imagem esperada é tipicamente preparada editando-se manualmente uma imagem de entrada.

De posse desses pares, pode-se então extrair os dados de treinamento. Para cada pixel, a partir da imagem de entrada constrói-se o vetor x de características e, este, juntamente com o valor y do mesmo pixel na imagem esperada, forma um exemplo de treinamento. Juntos, esses exemplos formam um conjunto de treinamento.

Consideramos o caso de segmentação como um problema de classificação pixel a pixel. Isto é, para cada pixel no mapa de uma imagem definimos uma categoria correspondente. Neste sentido, podemos dizer que a segmentação é um exemplo de um problema de **classificação**. Nesta categoria se encaixam os seguintes algoritmos:

- Redes Neurais
- Máquinas de Vetor de Suporte (SVMs)
- Classificadores Naive Bayes
- Árvores de decisão

2.2.2 Métricas de Performance ou Erro

Para o treinamento de um algoritmo de aprendizado se faz necessário o uso de métricas de distância. Através do processo de otimização desta métrica é que um modelo aprende os padrões em um conjunto de dados. Definimos como métrica uma função $d : X \times X \longrightarrow [0, \infty)$, comumente nominada como **distância**, que satisfaz as seguintes propriedades para todo $(x, y, z) \in D$:

- d não negativo: $d(x, y) \geq 0 \forall$
- identidade: $d(x, y) = 0 \iff x = y$
- d simétrico: $d(x, y) = d(y, x)$
- desigualdade triangular: $d(x, y) \leq d(x, z) + d(z, y)$

A escolha de uma métrica apropriada de avaliação de modelos é um dos tópicos de grande interesse na área de aprendizado de máquina. A decisão deve levar em consideração o tipo da tarefa, o conjunto de dados em qual etapa será utilizada a métrica.

Na literatura existem diferentes métricas possíveis para a avaliação de um modelo. Dentre elas, é importante mencionar as seguintes:

Acurácia

A acurácia é definida como a razão entre o número de predições e o número total de amostras, ou seja:

$$Acc = \frac{\#de\ predicoes\ corretas}{\#total\ de\ exemplos}$$

Esta métrica de classificação pode ser utilizada quando o conjunto de dados é balanceado, caso contrário esta métrica pode levar a um viés no modelo, uma vez que não temos controle sobre o tamanho da amostra para cada uma das classes possíveis.

Log loss

A métrica *log loss* busca penalizar as classificações falsas, através do cálculo utilizando a probabilidade para cada uma das classes. Suponha que existem N amostras e M classes, então vale :

$$LogLoss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} * log(p_{ij})$$

Onde:

1. y_{ij} indica se a amostra i pertence classe j ou não.
2. p_{ij} indica a probabilidade da amostra pertencer à classe j

Esta métrica é comumente utilizada para treinamento e avaliação de classificadores multiclasse.

Entropia Cruzada

A entropia cruzada nada mais é do que a aplicação da métrica *Log Loss* para o caso binário. Neste caso, a equação se reduz ao seguinte cálculo:

$$H_p = \frac{-1}{N} \sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i))$$

Esta métrica específica é utilizada para treinamento de classificadores com saída binária.

Precisão, Recall e Razão F1

Estas métricas são utilizadas como medidas de teste de classificadores binários, onde os testes são separados em quatro classes: Verdadeiros Positivos ou **TP**, onde o classificador classifica corretamente um caso verdadeiro, Falso Negativos **FN**, onde o classificador classifica uma amostra verdadeira como falsa, Falsos Positivos **FP**, onde o classificador classifica uma amostra falsa como verdadeira e Verdadeiros Negativos **TN**, onde o classificador classifica uma amostra falsa corretamente.

Em termos matemáticos definimos Precisão, Recall e F1 da seguinte maneira:

$$precisao = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{1}{\frac{1}{precisao} + \frac{1}{recall}}$$

Por consistir de uma média harmônica entre precisão e recall, a métrica F1 exibe uma boa avaliação sobre quão preciso é o classificador assim como quão robusto.

Precisões altas e recall baixo configuram um modelo de alta acurácia mas pouca generalização. Neste sentido, é desejado um modelo um modelo com alto $F1$

Erro Médio Absoluto (MAE)

O erro médio absoluto captura a média da diferença entre o conjunto retornado pelo modelo e o conjunto desejado, em termos absolutos. Definimos MAE como:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Apesar de retornar uma boa medida sobre o quão longínquo está a predição do conjunto verdade, esta métrica não retorna informações sobre a direção do erro, e é difícil calcular o gradiente dessa função. Este fato dificulta seu uso em algoritmos com *backpropagation.*

Erro Médio Quadrático (MSE)

Similar ao MAE, o erro médio quadrático captura a média da diferença entre o conjunto retornado pelo modelo e o conjunto desejado, com a diferença que este toma o quadrado do erro entre os dois. Definimos o MSE como:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Com relação ao MSE, esta métrica possibilita o uso do gradiente para minimizar o erro de sistema preditivo. Além disso, uma vez que o erro é elevado ao quadrado, esta métrica tende a valorizar erros maiores e desvalorizar erros menores.

2.2.3 Avaliação e Otimização Modelos

No fluxo padrão de um modelo de aprendizado de máquina, dividimos a avaliação de modelos em 3 etapas:

1. Treinamento
2. Validação
3. Teste

Em cada uma das avaliações, o erro calculado é utilizado de uma forma específica.

Treinamento

O erro de treinamento quantifica qual a distância entre a saída do modelo $f(\boldsymbol{x_{train}})$ e a saída esperada $\boldsymbol{y_{train}}$ no conjunto de treinamento $\boldsymbol{x_{train}}$ em relação ao conjunto esperado

y através de uma métrica de erro escolhida E_{train}. Este valor pode ou não ser utilizado para a otimização dos parâmetros do modelo, a depender se o problema em questão possui ou não uma distribuição conhecida. Em casos como regressão linear, a solução ótima pode ser calculada sem a necessidade do cálculo do erro, uma vez que a métrica SSR é convexa quando aplicada ao problema de regressão.

No capítulo 3.2 vamos mostrar como este erro é utilizado para otimizar os parâmetros de uma rede neural, utilizando a técnica de propagação para trás.

Validação

O erro de validação quantifica qual a distância entre a saída do modelo $f(\boldsymbol{x_{val}})$ e a saída esperada $\boldsymbol{y_{val}}$ no conjunto de validação $\boldsymbol{x_{val}}$ em relação ao conjunto esperado $\boldsymbol{y_{val}}$ através de uma métrica de erro escolhida E_{val}. Este valor é monitorado a fim de prevenir que o modelo tenha problemas de sobreajuste, ou seja, é com este valor que podemos verificar se em algum ponto perde-se a generalidade, de forma que o modelo fica muito específico ao conjunto de treino.

Teste

Similar aos demais, este erro quantifica qual a distância entre a saída do modelo $f(\boldsymbol{x_{test}})$ e a saída esperada $\boldsymbol{y_{test}}$ no conjunto de teste $\boldsymbol{x_{test}}$ em relação ao conjunto esperado $\boldsymbol{y_{test}}$ através de uma métrica de erro escolhida E_{test}. Este erro dita qual a performance final do modelo criado, feitas todas as otimizações planejadas pelo autor. **Importante**: O cálculo de erro de teste deve ser feito apenas uma vez, para ilustrar qual a performance do modelo criado. Feito isso, otimizações subsequentes devem ser performadas em outro conjunto de teste, caso contrário a métrica de teste perde o seu valor.

Capítulo 3

Redes Neurais

O termo **Redes Neurais** tem suas origens em tentativas de encontrar representações matemáticas de processamento de informação em sistemas biológicos (McCulloch and Pitts, 1943), neste artigo os autores modelam um neurônio como um interruptor que recebe impulsos de outros neurônios e, dependendo do peso total recebido é ativado ou permanece desativado. Este peso cuja entrada de outra célula é multiplicada corresponde a uma sinapse (conexões entre células nervosas). Em 1960 os autores mostraram que tais neurônios possuem propriedades similares ao cérebro: eles conseguiam identificar padrões complexos ainda que alguns neurônios fossem destruídos.

Mais tarde este mesmo conceito foi utilizado para leitura de texto por máquinas, em um trabalho que despertou o avanço da área de aprendizado de máquina, resultado em diversas pesquisas acadêmicas e aplicações diferentes que aplicavam o mesmo conceito.

3.1 Funções Feed-Forward

Matematicamente, o modelo mais básico dessa arquitetura pode ser descrito como uma série de transformações funcionais. Primeiro construímos M combinações lineares do vetor de entrada $\boldsymbol{x} = \{x_1, ..., x_D\}$:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

onde $j = 1, ..., M$ e (1) indica que os parâmetros estão na primeira 'camada' da rede. Além disso, vamos nos referir aos parâmetros w_{ji} como pesos e os parâmetros w_{j0} como viés. Os valores a_j são denominados ativações. Cada uma dessas ativações são transformadas utilizando uma função diferenciável e não-linear $h(.)$ retornando:

$$z_j = h(a_j)$$

Os valores z_j correspondem às unidades ocultas de uma das camadas da rede, configurando assim uma **camada oculta**. Estes valores então são combinados novamente para retornar

as unidades de ativação de saída:

$$a_k = \sum_{j=1}^{M} w_{jk}^{(2)} z_j + w_{k0}^{(2)}$$

onde $k = 1, ..., K$ e K é o tamanho de saída desejada. Finalmente, os valores a_k são transformados em valores de saída y_k utilizando uma **função de ativação** apropriada para o problema em questão.

Esta arquitetura é denominada **Totalmente Conectada**.

3.2 Backpropagation

O termo *Backpropagation* se refere ao algoritmo utilizado no treinamento de redes neurais (Hecht-Nielsen, 1992). O objetivo é calcular eficientemente o gradiente de uma função de erro $E(w)$ para posteriormente atualizar os parâmetros da rede de forma a minimizar o erro. A forma mais simples de realizar este procedimento foi introduzida por Rumelhart et al. (1986) e envolve o **gradiente descendente**.

O gradiente ∇f de uma função diferenciável f define um vetor com o sentido de maior crescimento da função f. Definimos $-\nabla f$ como o gradiente descendente da função f, naturalmente o gradiente descendente define um vetor com o sentido de maior decrescimento da função f. Uma vez que o objetivo é minimizar a função de erro, tomamos o gradiente descendente como sentido de atualização dos parâmetros de peso $\boldsymbol{w}$.

Como exemplo, podemos realizar a seguinte sequência de passos para otimizar uma rede *Feed-Fordward* com função de ativação $tanh$ e função de erro MSE:

1. Primeiro realizamos um passo de avaliação, calculando:

$$a_j = \sum_{i=0}^{D} w_{ij}^{(1)} x_i$$

$$z_j = tanh(a_j)$$

$$y_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

2. Feito isso, calculamos o gradiente para cada unidade de saída, utilizando:

$$\delta_k = y_k - t_k$$

3. Com esses valores calculamos o gradiente para cada unidade na camada oculta:

$$\delta_j = (1 - z_j^2) \sum w_{kj} \delta_k$$

4. Finalmente, calculamos as derivadas em relação aos pesos da primeira e segunda

camada, dados por:

$$\frac{\delta E_n}{\delta w_{ji}^{(1)}} = \delta_j x_i$$

$$\frac{\delta E_n}{\delta w_{kj}^{(2)}} = \delta_k z_j$$

Onde:

- y_k é o resultado da ativação da unidade k.
- t_k é o objetivo correspondente.
- t_i é um exemplo do vetor de entrada.

Repetindo esses passos um determinado número de vezes B retorna uma rede localmente otimizada para o conjunto de exemplos em questão.

3.3 Funções de Ativação

Funções de ativação são equações que determinam as ativações de nós de uma rede neural. Além de tomar a decisão de como passar um estímulo para as camadas subsequentes essas funções também são responsáveis por mapear as estímulos para intervalos menores, como o intervalo [0..1] no caso de um classificador. Essas funções devem ser computacionalmente rápidas de calcular, assim como suas derivadas, uma vez que durante o treinamento elas serão utilizadas milhares, ou até milhões de vezes para otimizar os parâmetros de uma rede.

Estas funções podem ser **lineares** ou **não-lineares**. O uso de funções não-lineares é recomendado em casos onde o conjunto de treinamento apresenta padrões complexos, uma vez que com a combinação dessas funções podemos aproximar funções lineares e não-lineares.

A seguir, daremos detalhes sobre as funções de ativação utilizadas nesse projeto.

3.3.1 Softmax

A função softmax unitária $\sigma : \mathbb{R}^k \longrightarrow \mathbb{R}^k$ é definida como:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j-1}^{K} e^{z_j}}$$

onde $i = 1, ..., K$

Tipicamente esta função é utilizada somente para camadas de saída, uma vez que ela é capaz de lidar com múltiplas classes e mapear uma saída normalizada entre 1 e 0, consistindo da probabilidade de que um exemplo da entrada seja de uma classe específica.

3.3.2 Rectified Linear Unit (ReLU)

A funcão relu é definida como a parte positiva de seu argumento, isto é:

$$relu(x) = max(0, x)$$

Estas funções geralmente são utilizadas em camadas de entrada e ocultas e são populares devido à eficiência de cálculo e a não-linearidade. Juntos, esses dois fatores fazem com que redes que utilizam essa função tenham rápida convergência e alta capacidade de capturar padrões.

3.4 Redes Neurais Convolucionais

Com respeito às características extraídas da imagem e que são utilizadas como entrada pelos algoritmos de aprendizado de máquina, muitas são computadas aplicando-se operações de convolução. Convoluções com determinadas máscaras são úteis, por exemplo, para realçar contornos.

A ideia de Redes Neurais Convolucionais surge com LeCun *et al.*, 1999 em uma tentativa de criar um modelo que é inerente a certas transformações na entrada. No caso de imagens por exemplo, podemos ter o interesse de ter uma mesma predição de classe para exemplos rotacionados, ou com leves distorções. Redes neurais tradicionais ignoram o fato de que pixels em uma vizinhança pequena compartilham mais informações entre si do que pixels distantes um do outro. Além disso, características que são úteis em uma região podem também ser úteis em outras regiões.

Essas noções são incorporadas por CNNs através de 3 mecanismos: campos receptivos locais, compartilhamento de pesos e sub-amostragem. Em uma camada convolucional as unidades são organizadas em planos chamados de mapa de features. Cada unidade em um mapa recebe informações apenas de uma região pequena da imagem.

Se interpretarmos as unidades como detectores de característiscas, então todas as unidades em um mesmo mapa detectam um mesmo padrão, mas em locais diferentes da imagem de entrada. Devido ao compartilhamento de pesos, a avaliação das ativações dessas unidades é equivalente a uma convolução da intensidade de cor do pixel da imagem com um *kernel* contendo os parâmetros de pesos.

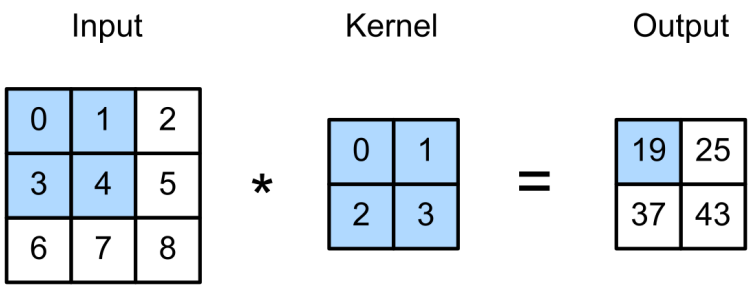

Figura 3.1: *Um exemplo de convolução.*

3.5 Operações de Convolução

Convolução é uma operação que expressa a quantidade de sobreposição entre duas funções f e g. Em termos matemáticos, definimos a convolução $f * g$ como:

$$[f * g](t) = \int_0^t f(\tau)g(t-\tau)\delta\tau$$

Em processamento de imagens no entanto, operações de convolução são utilizadas para diferentes tarefas como borrar, realçar detalhes e detectar contornos de objetos. Para isso, são definidas matrizes denominadas como kernel e é realizada a convolução pixel a pixel, da imagem com o kernel escolhido seguindo a equação:

$$g(x,y) = w * f(x,y) = \sum_{s=-a}^{a}\sum_{t=-a}^{b} w(s,t)f(x-s,y-t)$$

onde w é o kernel, $g(x,y)$ é a imagem resultante e $f(x,y)$ é a imagem original.

Note que o resultado da convolução depende unicamente do kernel utilizado. Para o pré-processamento de imagens existem uma série de kernels que podem ser obtidos através de equações matemáticas.

3.6 Redes Neurais Totalmente Convolucionais

Uma das formas de segmentar imagens de forma rápida e computacionalmente eficiente é através de redes Redes Neurais Totalmente Convolucionais ou **FCNs**. Estas redes consistem em suscetivas transformações convolucionais aplicadas à janelas de pixel W que têm como objetivo, ou conjunto verdade, uma janela de pixels denominada *patch*.

Como Shelhamer *et al.*, 2016 descreve em seu artigo "Fully Convolutional Networks for Semantic Segmentation", cada camada em uma FCN consiste de um vetor tridimensional de tamanho $h \times w \times d$ onde h e w são dimensões espaciais e d é a dimensão do canal ou *feature*. Diferentemente de CNNs para segmentação padrão, FCNs não utilizam-se de técnicas de upsampling, nem de arquiteturas no estilo *bottleneck*. Aqui, tomamos vantagem do fato de que sub-amostras (janelas) da imagem podem ser utilziadas como conjunto de treino para a rede neural, e assim podemos fazer a segmentação da imagem inteira através da segmentação dessas janelas retiradas da imagem, o que resulta em tempos de processamento melhores e menor quantidade de memória alocada.

Capítulo 4

Metodologia

Como descrito no título, podemos dividir este trabalho duas grandes etapas: **Segmentação** e **Síntese**. Para realizar a segmentação de imagens, treinamos diferentes classificadores de mesma arquitetura mas com conjuntos de treino (campos receptivos) diferentes. Estes campos receptivos são janelas de pixels com centro deslocado em alguns pixels em relação ao seu objetivo.

Ao realizar este deslocamento, conseguimos capturar um campo receptivo maior, sem perdas de generalidade. Contudo, se faz necessário um mecanismo de síntese de saídas, uma vez que cada classificador dará um resultado diferente para um mesmo exemplo de entrada. Neste sentido, exploramos diferentes maneiras de combinar estes classificadores.

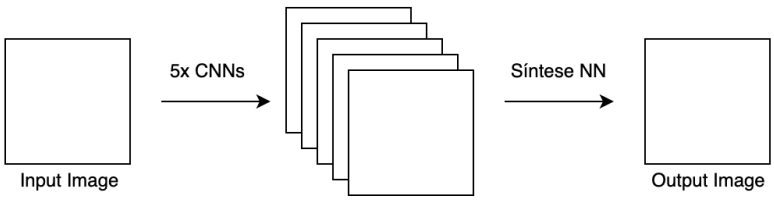

Figura 4.1: *Esquema final desejado.*

No sentido de implementação utilizamos as definições disponíveis pela biblioteca Keras, utilizando Tensorflow como plataforma *backend*.

4.1 Segmentação

Neste trabalho utilizamos redes convolucionais com arquitetura FCN para realizar o trabalho de segmentação, utilizando Softmax como camada de ativação. Cada rede toma como entrada uma janela K de pixels de tamanho 27×27 e um canal de cor, e mapeia um conjunto de pixels M de tamanho 5×5, restrito ao conjunto binário $\{0, 1\}$.

Considerando C como o número total de classificadores e $c_{i,j}$ como a classe real de um pixel p na posição (i, j), a responsabilidade de cada um destes classificadores é atribuir

uma classe $\hat{c}_{i,j}$ para o pixel $p_{i,j}$. Devido ao fato de que utilizamos Softmax como camada de ativação, $\hat{c}_{i,j}$ pode ser obtido através da seguinte operação:

$$\hat{c}_{i,j} = \begin{cases} 0 & \text{se } s_{i,j} < 0.5 \\ 1 & \text{caso contrário} \end{cases} \tag{4.1}$$

Onde $s_{i,j}$ denota a saída da rede, contendo a probabilidade de um pixel pertencer à classe 1.

Com isso, podemos resumir esta etapa como uma função:

$$S_g : \mathbb{Z}^{27\times27}_{0\leq x\leq 255} \longrightarrow \mathbb{R}^{5\times5\times C}_{0\leq x\leq 1}$$

Esta função é aplicada para todas as janelas amostradas a partir da imagem inicial, gerando assim um conjunto de C matrizes de dimensão igual à imagem inicial, contendo as probabilidades de cada pixel pertencer à classe 1.

4.1.1 Arquitetura de Segmentação

No sentido de arquitetura de redes neurais, utilizamos 5 camadas internas de convolução, com kernel de tamanho 3×3 e com ativações do tipo relu. Utilizamos entropia cruzada como medida de otimização e *learning rate* adaptativo (Zeiler, 2012). A saída da rede de classificação utilizada foi *Softmax*. Esta função nos permite extrair o grau de confiança do modelo em uma predição, que mais tarde será utilizado para realizar o voto majoritário. Mais detalhes de arquitetura podem ser consultados na figura abaixo.

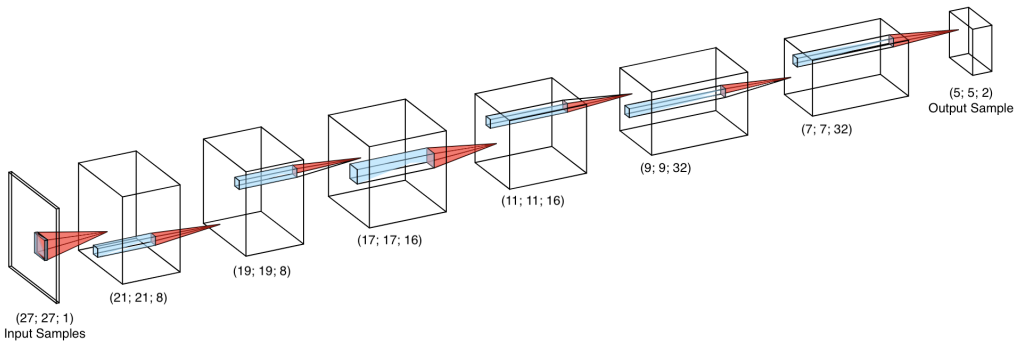

Figura 4.2: *Rede convolucional de segmentação.*

4.2 Síntese

Podemos definir como função de síntese uma função $Si : \mathbb{R}^C \longrightarrow \mathbb{Z}_{0,1}$. Onde C é o número de classificadores. Ao aplicar essa função pixel a pixel em cada um dos vetores

gerados pela sobreposição das imagens das janelas de saída de redes, podemos criar a seguinte abstração de classificação:

$$S_g \circ S_i \ : \ \mathbb{Z}_{0 \leq x \leq 255}^{27 \times 27} \longrightarrow \mathbb{Z}_{0,1}^{5 \times 5}$$

Com isso, basta aplicar essa função para todas as janelas amostradas a partir da imagem inicial para ter o resultado final de uma imagem segmentada.

Existem diferentes formas de criar tais funções de síntese S_i. Neste trabalho exploramos os seguintes: voto majoritário, voto ponderado e rede neural de síntese.

4.2.1 Voto Majoritário

Como o nome sugere, o voto majoritário consiste em atribuir como classe final ao pixel a classe mais votada entre os classificadores para aquele pixel, ou seja:

$$M = \sum_{n=0}^{N} \hat{\mathbf{c}}_{\mathbf{i},\mathbf{j_n}}$$

$$S_{i_m}(\mathbf{c_{i,j}}) \begin{cases} 0 & \text{se } M/N < 0.5 \\ 1 & \text{caso contrário} \end{cases} \tag{4.2}$$

Onde $\hat{\mathbf{c}}_{\mathbf{i,j}}$ denota o vetor de classes atribuídas para o pixel $p_{i,j}$ pelos classificadores através do procedimento descrito na equação **4.1** e N denota o número de classificadores.

Note que em casos de empate no número de votos de classes este classificador arredonda o valor para a classe 0, e este procedimento só faz sentido em casos de classificadores binários. Para evitar empates neste trabalho empregamos um número ímpar de classificadores em uma tarefa com duas classes.

4.2.2 Voto Ponderado

O voto ponderado toma vantagem do fato de que os classificadores possuem um grau de confiança $\mathbf{s}_{\mathbf{i,j}}$ na predição. Com essa informação podemos retornar como resposta a classe com maior grau de confiança, isto é:

$$P = \sum_{n=0}^{N} \hat{\mathbf{c}}_{\mathbf{i},\mathbf{j_n}} * \mathbf{s}_{\mathbf{i},\mathbf{j_n}}$$

$$S_{i_p}(\mathbf{c_{i,j}}) \begin{cases} 0 & \text{se } P < 0.5 \\ 1 & \text{caso contrário} \end{cases} \tag{4.3}$$

Novamente, este procedimento só faz sentido em casos de classificadores binários

4.2.3 Rede Neural de Síntese

Para utilizar uma rede neural de síntese definimos uma rede neural no estilo *Fully Connected* com número de canais de entrada igual ao número de classificadores e número de canais de saída igual ao número de classes desejadas. Naturalmente, existem diferentes arquiteturas que podem ser utilizadas para realizar este procedimento, e devem ser levados em conta o número de classes, o número de exemplos e a quantidade de classificadores. Neste trabalho, utilizamos apenas uma camada interna, com 10 tensores.

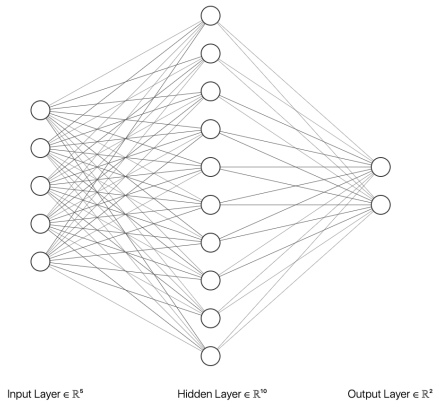

Figura 4.3: *Rede neural de Síntese.*

Ao utilizar uma rede neural de síntese é possível identificar padrões de síntese de redes, isto é, se torna possível que o voto ponderado em um pixel não seja o resultado final da classe naquele pixel, devido ao fato que determinadas combinações de saída de classificadores podem ser mais relevantes que outras.

Sobre as ativações, utilizamos relu e entropia cruzada nesta rede. A função de saída utilizada foi *Softmax*, porém podemos utilizar função de saída sem perda de generalidade, uma vez que estamos interessados somente em qual classe será atribuida para o conjunto de pixels proveniente da fase de classificação. Desta vez utilizamos *adam* (Kingma e Ba, 2014) como otimizador de *learning rate*.

Capítulo 5

Resultados

Aplicamos a arquitetura definida na metodologia em exemplos de segmentação de quadrinhos. Cada um dos exemplos de entrada constitui uma página de tamanho 1170×827 escaneada de uma revista de quadrinhos mangá. O conjunto verdade é constituído por uma imagem de tamanho de tamanho 1170×827 com pixels de valor binário: aqueles que fazem parte de uma representação de texto recebem label 1 e o restante recebe label 0. Para simplificar o processo, estas páginas foram transformadas em tons de cinza antes de serem processadas pelo modelo de segmentação.

Ao total, utilizamos 5 classificadores para a segmentação, sendo que cada um deles teve um deslocamento em relação ao *patch* objetivo:

1. **Sup Left**: O objetivo fica no canto superior esquerdo da janela.
2. **Sup Right**: O objetivo fica no canto superior direito da janela.
3. **Bot Left**: O objetivo fica no canto inferior esquerdo da janela.
4. **Bot Right**: O objetivo fica no canto inferior direito da janela.
5. **Center**: O objetivo fica no centro da janela.

Quando combinadas, as janelas capturam um campo receptivo de tamanho total 41×41 para um mesmo objetivo de tamanho 5×5.

O conjunto de treino dos modelos de segmentação consistiu de um conjunto de 15 imagens do dataset, separadas em janelas de tamanho 27×27. Validamos este modelo utilizando amostras de 5 imagens. Para treinar o modelo de síntese, avaliamos 15 imagens com os classificadores, concatenamos e utilizamos como conjunto de treino os vetores de tamanho 1×5 contendo cada uma das predições dos classificadores para um mesmo pixel.

A acurácia média final no conjunto de validação dos classificadores foi 0.994 enquanto que da rede neural de síntese foi 0.991, isto é, em aproximadamente 99% dos casos o pixel dado pela rede de classificação foi correto e o mesmo vale para a rede de síntese.

Treinadas ambas as redes, utilizamos um conjunto de 30 imagens para avaliar cada um dos classificadores. O resultado foi o seguinte:

Model List		
Model	LogLoss	Acc
Sup Left	234.1425	0.9667
Sup Right	876.3428	0.930
Bot Left	288.8897	0.9668
Bot Right	107.5662	0.9834
Center	101.3952	0.9719
Weig Avg	290.615	0.987
Neur Net	**81.175**	**0.985**

Tabela 5.1: *Modelos e métricas correspondentes no conjunto de teste.*

Em termos de Erro Logarítmico podemos perceber pela tabela que, em média, a rede neural de combinação atingiu um erro equivalente a pouco **menos de um terço** da média das redes neurais sozinhas. Este resultado contudo, não se reflete totalmente na acurácia do modelo. Em termos de acurácia na predição de pixels, o modelo de combinação de redes ficou em segundo lugar, logo após o de voto ponderado. Esta performance contudo, supera a média dos outros sistemas de classificação em uma média de 2.5% de acurácia. Esta alteração se reflete na segmentação. Como podemos ver na figura, o modelo de segmentação conseguiu abstrair melhor as figuras quando comparado com um dos modelos.

Além disso, também foi possível ver que a performance cai quando alteramos o peso entre as classes. Isto provavelmente é resultado do fato de que, ao aumentar o peso da classe 1 (Branca) melhoramos a visibilidade do texto na imagem, mas também aumentamos o ruído no resultado. Neste trabalho não foi possível explorar maneiras de aumentar a qualidade do texto sem aumentar o ruído.

A **Figura 4** ilustra a distribuição de acurácia para o conjunto de testes. Através dela, podemos concluir que o modelo com combinação de resultados obteve resultados consistentemente melhores que os modelos individuais.

Capítulo 6

Conclusão

A área de aprendizado de máquina ainda possui muitos temas não explorados, que podem render ótimos resultados para o campo acadêmico. Neste trabalho conseguimos explorar um destes temas, utilizando técnicas comuns como CNNs e Redes Neurais para chegar a um resultado melhor do que algumas das técnicas empregadas hoje no espaço de segmentação de imagens, sem a necessidade de empregar arquiteturas de aprendizado profundo. O tempo de execução e eficiência de algoritmos de aprendizado de máquina vem se tornando componente essencial para que se torne possível a aplicação destes modelos em tarefas do dia a dia.

Particularmente, este é um ótimo exemplo de como podemos compor diferentes algoritmos de aprendizado de máquina em partes de uma mesma tarefa. Este desacoplamento nos permitiu inserir novos elementos de complexidade que geraram melhores resultados. É fato particular que a melhora do erro logarítmico médio no conjunto de testes não resultou necessariamente em uma melhora da acurácia média. Contudo, é natural imaginar que a síntese por meio de redes neurais teria um melhor erro, visto que durante o treino essa é a métrica utilizada para otimizar o modelo. O uso de métricas diferentes para treino de ambas as arquiteturas de rede não foi explorado neste trabalho, mas é algo que pode gerar performances ainda melhores daquelas que foram reportadas aqui.

Dentro deste mesmo trabalho ainda restam alguns tópicos interessantes e que podem gerar novas pesquisas. Aqui foi possível perceber que existe uma relação direta entre o peso atribuído às classes e a qualidade da síntese. Além disso, também percebemos que classificadores com campos receptivos levemente deslocados podem ter performance melhor em alguns casos, e que a combinação de tais classificadores é tarefa de certa complexidade e que requer estudos para melhor entender quais as melhores técnicas e maneiras de realizar a tarefa. Seguindo com este estudo ainda se faz necessário juntar as duas redes neurais em um mesmo pipeline, de forma a otimizar o processo de segmentação fim-a-fim.

Finalmente, agradeço à todos aqueles que contribuíram para que este trabalho fosse um sucesso. Particularmente aos orientadores deste trabalho que demonstraram dedicação e conhecimento no assunto sempre que requisitados, à Amazon por prover os recursos computacionais necessários e ao Instituto de Matemática e Estatística da USP por disponibilizar

os espaços de estudo e trabalho utilizados durante o ano.

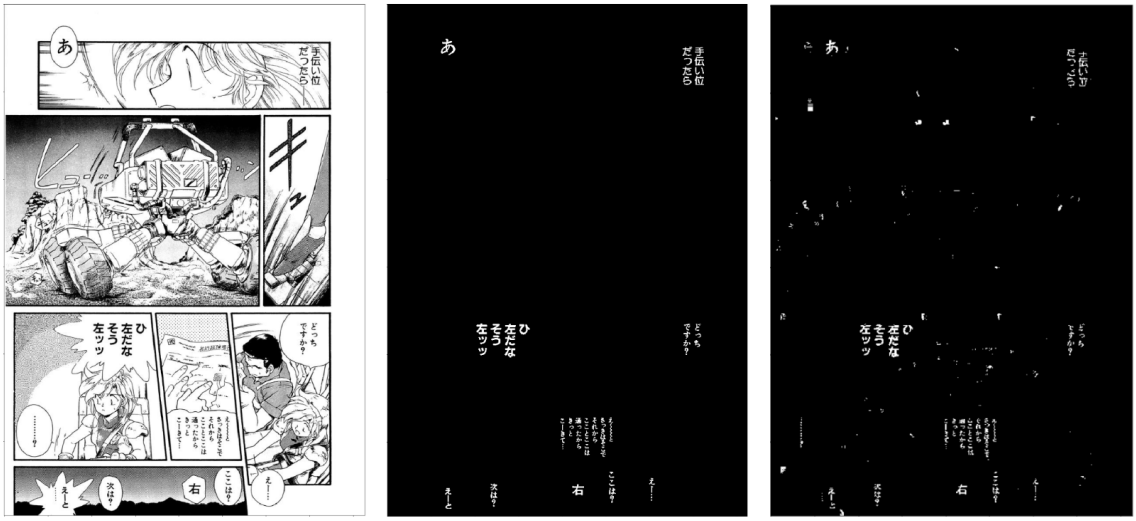

Figura 1: *Resultado final do trabalho: Entrada à esquerda, Ground Truth ao centro e saída da combinação de redes à direita.*

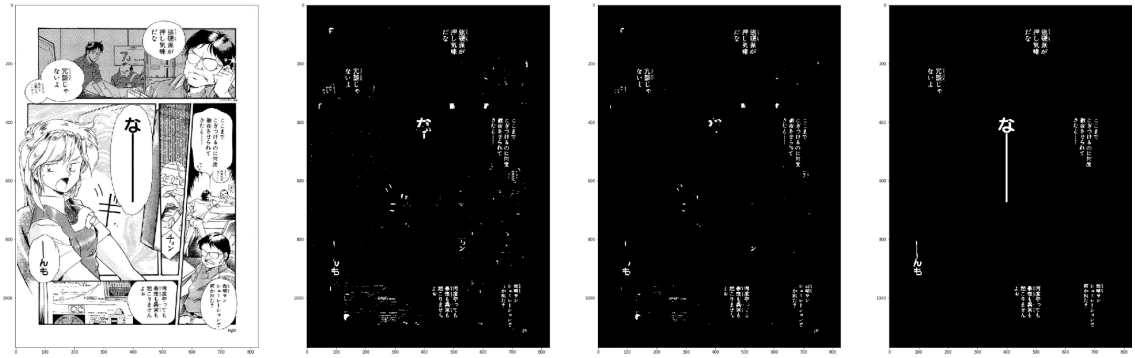

Figura 2: *Entrada e saída dos modelos: À esquerda temos o quadrinho e a direita o Ground Truth. A segunda imagem representa a saída esquema de Redes Neurais e a terceira imagem a saída obtida através do método de voto ponderado*

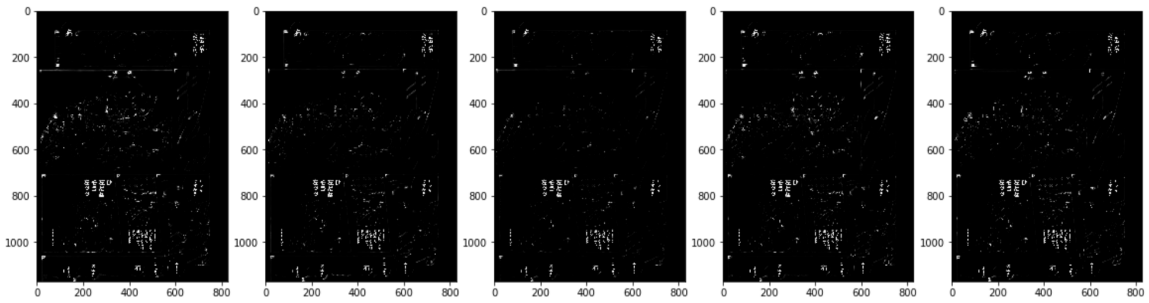

Figura 3: *Resultados das redes individuais aplicados a um exemplo de imagem.*

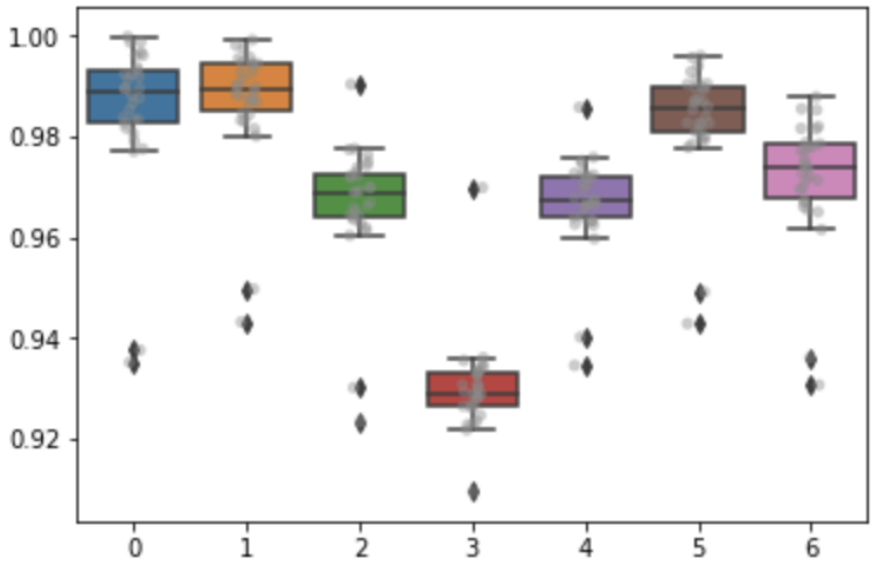

Figura 4: *Distribuição de acurácias de cada um dos modelos. Da esquerda para a direita temos: Rede Neural, Voto Ponderado e os classificadores individuais na ordem descrita na seção 5.*

Referências

[Hecht-Nielsen 1992] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". Em: *Neural networks for perception*. Elsevier, 1992, pgs. 65–93 (citado na pg. 12).

[Hirata 2008] Nina ST Hirata. "Multilevel training of binary morphological operators". Em: *IEEE Transactions on pattern analysis and machine intelligence* 31.4 (2008), pgs. 707–720 (citado na pg. 2).

[Kingma e Ba 2014] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG] (citado na pg. 20).

[LeCun *et al.* 1999] Yann LeCun, Patrick Haffner, Léon Bottou e Yoshua Bengio. "Object recognition with gradient-based learning". Em: *Shape, contour and grouping in computer vision*. Springer, 1999, pgs. 319–345 (citado na pg. 14).

[Lu e Shridhar 1996] Yi Lu e Malayappan Shridhar. "Character segmentation in handwritten words—an overview". Em: *Pattern recognition* 29.1 (1996), pgs. 77–96 (citado na pg. 1).

[Mustaqeem *et al.* 2012] Anam Mustaqeem, Ali Javed e Tehseen Fatima. "An efficient brain tumor detection algorithm using watershed & thresholding based segmentation". Em: *International Journal of Image, Graphics and Signal Processing* 4.10 (2012), pg. 34 (citado na pg. 1).

[Shelhamer *et al.* 2016] Evan Shelhamer, Jonathan Long e Trevor Darrell. "Fully convolutional networks for semantic segmentation". Em: *PAMI* (2016) (citado na pg. 15).

[Solomonoff 1957] Ray Solomonoff. "An inductive inference machine". Em: *IRE Convention Record, Section on Information Theory, Part 2* (1957), pgs. 56–62 (citado na pg. 6).

[Zeiler 2012] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: 1212.5701 [cs.LG] (citado na pg. 18).