Análise de Sentimentos em Textos em Português do Brasil Usando Redes Neurais Convolucionais

Lucas Alves Sobrinho[1], Alceu de Souza Britto Junior[2]

[1]Departamento de Informática – Universidade Estadual de Ponta Grossa (UEPG)
Avenida General Carlos Cavalcanti, 4748
CEP 84030-900 – Ponta Grossa, PR – Brasil

[2]Programa de Pós-Graduação em Informática (PPGIa) – Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155
CEP 80215-901 – Curitiba, PR – Brasil

lasobrinho7@gmail.com, alceu@ppgia.pucpr.br

Abstract. *Written text classification through deep learning recently has become an important study subject for many researchers in the industry. Traditional machine learning techniques, such as Support Vector Machines (SVM), have been outperformed by techniques that uses Convolutional Neural Networks (CNNs). Through this method it is possible to classify, with high precision and speed, texts based on two or more labels that identify the text nature, for instance negative or positive sentences. This article presents a deep learning architecture based on CNN to train sentiment classifiers for texts in Brazilian Portuguese.*

Resumo. *A classificação de textos escritos através de aprendizado profundo tornou-se recentemente um importante tópico de estudo por parte de muitos pesquisadores. Técnicas tradicionais de aprendizagem de máquina para classificação, tais como o Support Vector Machine (SVM), têm sido superadas por técnicas que se apoiam no uso Redes Neurais Convolucionais (CNNs). Através desse método pode-se classificar, com alta precisão e velocidade, textos de acordo com dois ou mais rótulos identificando a natureza do texto, por exemplo sentimentos negativos ou positivos. Este artigo apresenta uma arquitetura de aprendizagem profunda com base em CNN para o treinamento de classificadores de sentimentos em textos no idioma Português do Brasil.*

1. Introdução

Nos dias de hoje, diversos tipos de dados estão disponíveis de maneira livre nas redes públicas de computadores, em especial a Internet. A melhoria da acessibilidade da população a Internet, aos computadores e aos dispositivos móveis estimula a criação e compartilhamento de conteúdo online, em especial o conteúdo em forma de texto escrito. Esses fatores transformam a Internet em uma grande fonte para extração de textos escritos em diversas línguas. Dentre os diversos tipos de informações que podem ser extraídos destes dados disponíveis online estão os *sentimentos* ou as *opiniões* que as pessoas expressam em relação a um assunto específico. Pode-se, por exemplo, extrair

informações de quão boa está a reputação de uma empresa, opiniões positivas ou negativas de consumidores em relação a produtos disponíveis no mercado, sentimentos de reações de leitores em relação a notícias ou pesquisas, dentre uma infinidade de aplicações em diversas áreas de interesse. Embora esses tipos de informações possam ser extraídos usando métodos tradicionais manuais, como pesquisas de opinião ao público, o foco deste trabalho é a extração automática, precisa e eficaz de sentimentos em textos [Yi et al. 2003].

A análise e extração de sentimentos e opiniões é, porém, uma tarefa difícil de ser automatizada, requerendo entendimento do contexto das sentenças, do senso comum para interpretação dos textos e de conhecimento avançado da teoria linguística do idioma sendo usado. A análise de sentimentos e opiniões em sentenças gera discussões até mesmo entre humanos. Quando se tenta determinar uma decisão sobre a positividade ou negatividade de frases em texto é comum que, quando apresentadas a um grupo pequeno de pessoas, não se consiga entrar em um consenso final, gerando discussão sobre as sentenças analisadas [Nasukawa e Yi 2003]. Além disso, adiciona-se o fato de muitas sentenças, especialmente aquelas relacionadas a opiniões sobre um assunto, estarem desfiguradas por conta de expressões ou construções irônicas, o que dificuldade uma análise exata da opinião ou sentimento sendo expressado, mesmo sob a análise manual por humanos em vez do uso de computadores com métodos automáticos.

Este trabalho tem como principal interesse o estudo da automatização do processo de análise e extração de sentimentos, usando uma abordagem baseada em aprendizagem de máquina, embora existam outras abordagens mais rudimentares com maior necessidade de intervenção humana na modelagem da solução. A abordagem de aprendizagem de máquina para análise de sentimentos é baseada no uso dos algoritmos disponíveis na literatura, aplicando-os como um problema convencional geral de classificação de textos que se utiliza de características (*features*) linguísticas ou sintáticas. De acordo com Medhat et al. (2014), em processamento de linguagem natural as técnicas de aprendizagem de máquina podem ser divididas em aprendizado supervisionado ou não-supervisionado. Neste trabalho é explorado o método de aprendizagem supervisionado, o qual depende do uso de textos previamente classificados com os rótulos ou classes corretas. Esse conjunto de textos rotulados é então usado para o treinamento do classificador. Existem diversas técnicas de aprendizado supervisionado na literatura, sendo algumas delas bem consolidadas, tendo como exemplo os classificadores probabilísticos (*Naïve Bayes Classifier*, *Bayesian Networks*, *Maximum Entropy Classifier*, dentre outros) e os classificadores lineares, como a *Support Vector Machine Classifier* (SVM) e as Redes Neurais (NNs) convencionais [Medhat et al. 2014].

Embora todos os métodos citados anteriormente já tenham sido testados e validados na literatura, conforme mostra Medhat et al. (2014), o foco deste trabalho é o uso de um método não convencional para a classificação de textos. Para a análise de sentimentos em textos neste trabalho será usado uma abordagem supervisionada de aprendizagem profunda (*deep learning*) baseada em Rede Neural Convolucional (CNN). As CNNs, inventadas inicialmente para uso no reconhecimento de objetos em imagens digitais e na classificação dessas imagens, recentemente têm sido fortemente consideradas para o processamento de linguagem natural e classificação de textos, após diversos pesquisadores terem obtido resultados notáveis e desempenho considerável

quando comparado a técnicas tradicionais de classificação [Zhang e Wallace 2015]. A arquitetura das CNNs modernas, otimizada por LeCun (1998a), utiliza camadas com filtros de convolução que são aplicados sobre as entradas da rede. Os filtros extraem as características mais importantes contidas no vetor de entrada, para depois serem processadas, transformadas em um vetor de saída e categorizadas no fim da rede por algum método de classificação tradicional [LeCun et al. 1998a].

As Redes Neurais Convolucionais foram concebidas para trabalharem sobre entradas de dados em formato de matrizes, como exemplo uma imagem com um ou mais canais de cores. Para que uma CNN possa trabalhar sobre dados textuais é necessário a conversão da sentença ou do texto para um formato de matriz com um número finito de linhas e colunas, fazendo com que os dados textuais se pareçam com um formato esperado pela CNN. Neste trabalho será utilizado o conceito de *word embedding*, onde um vetor de palavras ou frases é convertido em um vetor de números reais de tamanho fixo. Após este procedimento, o vetor de números pode então ser passado como vetor de entrada para a CNN e então ter os sentimentos extraídos das sentenças ou textos [Kalchbrenner et al. 2014].

Para que se possa obter o vetor de entrada esperado pela CNN e para que seja possível treiná-la é necessário ter uma base de dados textuais. O fato de se ter facilidade de acesso a textos disponíveis em diversas fontes distribuídas pela Internet faz com que seja possível a extração automática e armazenamento destes dados textuais. Com isso pode-se manter um banco de dados contendo informações de texto criadas manualmente por pessoas, representando assim uma compilação de exemplos válidos de textos de um ou mais idiomas [Nasukawa e Yi 2003]. Para a etapa de treinamento da CNN é importante que, além da extração automática de textos, também se faça a obtenção do tipo de reação de sentimento das pessoas em relação a cada texto da base de dados. Com isso cria-se pares de texto e informações de sentimento válidas, gerando assim uma base de dados (*dataset*) de textos e sentimentos para posterior uso em sistemas de aprendizagem profunda, como as Redes Neurais Convolucionais.

Neste trabalho é treinada uma Rede Neural Convolucional simples inspirada no trabalho de Kim (2014). Esta CNN implementa uma camada de convolução usando vetores de palavras obtidos através da técnica de *word embedding*, tendo como saída do sistema a predição do sentimento do texto sendo analisado. A distribuição das seções deste trabalho é apresentada a seguir. Na Seção 2 é descrito brevemente o funcionamento de uma Rede Neural Convolucional. A Seção 3 apresenta conceitos relativos a *word embedding* e sua relação com a camada de entrada da CNN. Na Seção 4 é discutido detalhes específicos da arquitetura e do modelo de Rede Neural Convolucional a ser utilizada neste trabalho, tais como os operadores relevantes e as camadas da CNN sendo proposta. Na Seção 5 é feita uma discussão a respeito do *dataset* utilizado e dos experimentos realizados com a CNN. Por fim, nas Seções 6 e 7 são realizados, respectivamente, a discussão dos resultados e a conclusão do trabalho.

2. Classificação com Redes Neurais Convolucionais

As Redes Neurais Convolucionais, através do trabalho de LeCun (1998a), obtiveram grande adoção por parte de diversos pesquisadores da área para resolver vários problemas de visão computacional com a técnica de aprendizado profundo. De maneira

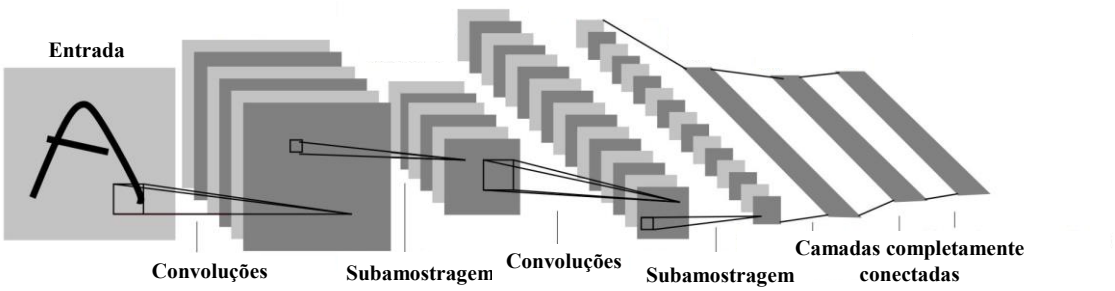

Figura 1. Arquitetura e camadas básicas de uma CNN [LeCun et al. 1998a]

similar às redes neurais artificiais convencionais, as CNNs também fazem uso de unidades chamadas de neurônios, os quais possuem pesos e *bias* numéricos que são aprendidos e aprimorados durante o processo de treinamento da rede. Também fazem uso de uma função de perda (*loss function*, tendo como exemplo o algoritmo *softmax* [Bridle 1990a]) nas suas últimas camadas e o conceito de que a rede completa representa uma função diferenciável de classificação continua sendo válido. Entretanto, diferentemente de redes neurais convencionais, CNNs esperam como entrada vetores de tamanho fixo, como imagens, por exemplo, e identificam padrões nessa entrada. Também é possível utilizar outros tipos de informações em uma CNN convencional, como áudio ou textos escritos, desde que a representação desses dados seja em formato de um vetor de tamanho fixo.

A grande vantagem das CNNs é o fato de funcionar a partir do conceito de que alguns pesos e *biasses* são compartilhados, reduzindo drasticamente o número de parâmetros que podem ser aprendidos e fazendo com que o treinamento seja mais eficiente, diminuindo assim os requisitos computacionais necessários para que a rede seja treinada [LeCun et al. 1998a]. Tal compartilhamento de parâmetros é possível porque a CNN utiliza a técnica de campos receptivos locais, os quais permitem realizar a operação de convolução de filtros sobre as camadas anteriores da rede, gerando assim mapas de características relativos àquele filtro. Outra técnica utilizada por CNNs é a subamostragem (*subsampling* ou *pooling*), onde os mapas de características são simplificados ou reduzidos de tamanho. Por fim, as características (*features*) geradas pelas camadas anteriores da CNNs são agrupadas e passadas por uma camada completamente conectada de neurônios (como observado em redes neurais convencionais) para que a entrada seja classificada em um ou mais rótulos previamente definidos. Estes três componentes básicos são mostrados na Figura 1 e serão explicados com maior detalhe nas subseções seguintes.

2.1. Camada de Convolução

A camada de convolução de uma CNN tem como objetivo extrair características ou padrões dos vetores de entrada. Isso é feito através de uma operação de convolução sobre um vetor de entrada, onde um filtro de tamanho fixo é iterado sobre todos os elementos da entrada. Essa iteração completa gera um outro vetor, chamado de mapa de características (*feature map*). Na analogia com campos receptivos locais, o filtro de iteração representa o processamento em apenas uma parte da entrada, extraindo assim a característica mais marcante daquela região e registrando-a no mapa de características

na sua respectiva posição. A camada de convolução em uma CNN consiste de vários mapas de características, sendo que cada mapa possui seu próprio filtro de convolução distinto, a fim de extrair características diferentes do vetor de entrada [LeCun et al. 1998a]. O filtro de convolução é composto por um número fixo de coeficientes (pesos) treináveis e mais um valor de *bias*, também treinável. Matematicamente, na operação de convolução, os valores numéricos dos elementos do vetor de entrada estão sujeitos a uma operação de produto escalar, composta da multiplicação pelo coeficiente e uma adição com o valor de *bias*, conforme mostra a equação abaixo, para um mapa de características *k*:

$$z_{i,j,k} = w_k x_{i,j} + b_k \qquad \text{[Eq. 1]}$$

onde *z* representa a característica calculada no ponto (i, j), *w* e *b* são, respectivamente, os coeficientes e *bias* do filtro de convolução do mapa de características *k*, e *x* é o elemento do vetor de entrada na posição (i, j) [Gu et al. 2015].

Após calculado o mapa de características para cada um dos filtros da camada de convolução, a característica z é passada para uma função de ativação, para que sejam introduzidas não-linearidades à CNN com o objetivo de detectar características não-lineares. O valor de ativação é calculado conforme a equação:

$$v_{i,j,k} = a(z_{i,j,k}) \qquad \text{[Eq. 2]}$$

onde *v* representa o valor de ativação após ser calculado pela função de ativação *a*, tendo como entrada o valor de característica *z* calculado anteriormente através do filtro de convolução [Gu et al. 2015]. Existem diversas funções de ativação na literatura, sendo mais comuns em CNN o uso da *sigmoid*, *tanh* [LeCun et al. 1998b] e *ReLU* [Nair e Hinton 2010].

2.2. Camada de Subamostragem

Outra vantagem importante das CNNs é a sua invariabilidade com relação a distorções das características extraídas. Essa invariabilidade é obtida com a operação de subamostragem ou *pooling*. O que essa operação faz é reduzir a precisão de um mapa de características diminuindo as dimensões do vetor. No caso de imagens, essa operação é equivalente a uma redução de resolução. Ao reduzir a resolução do mapa de características se reduz também a sensibilidade a distorções das características a serem identificadas [LeCun et al. 1998a]. Na detecção de objetos em imagens, essas distorções são representadas por rotações, translações e escala dos objetos na imagem de entrada. Na classificação de sentenças, a distorção pode ser o posicionamento de palavras ou expressões em uma sentença ou texto, bem como a falta de certas a palavras de ligação numa frase ou também a escrita incorreta de alguns termos. A operação de subamostragem pode ser definida por:

$$y_{i,j,k} = p(v_{i,j,k}) \qquad \text{[Eq. 3]}$$

onde *y* é o novo valor computado pela função de subamostragem *p*, a qual recebe um valor de ativação *v* como entrada, calculado anteriormente [Gu et al. 2015].

São operações comuns de subamostragem o *average pooling* [LeCun et al. 1998a e Wang et al. 2012], *max pooling* [Boureau et al. 2010] e mais recentemente o *k-max pooling* e *dynamic k-max pooling* [Kalchbrenner et al. 2014]. Além de ser

Figura 2. Exemplo de uma matriz de sentenças. Adaptado de Kim (2014).

importante para a redução da sensibilidade ao detectar *features*, a operação de subamostragem também reduz a complexidade computacional do problema ao reduzir as dimensões dos vetores ao longo da rede neural convolucional.

Numa configuração normal de CNN, a camada de convolução é tipicamente seguida de uma camada de subamostragem, podendo esse par de camadas ser usado repetidas vezes ao longo da rede. Isso possibilita que as primeiras camadas da rede identifiquem características de baixo nível, tais como arestas e curvas, ao passo que as camadas mais distantes do início da rede identificam características mais abstratas de nível mais alto, por conta do uso dos filtros de convolução e das várias operações de subamostragem.

2.3. Camada Completamente Conectada

A última camada de uma CNN é a camada completamente conectada de neurônios (*fully connected layer*). Ela é responsável por receber como entrada os valores computados pelas camadas de convolução e subamostragem e calcular os valores de classificação (ou *scores*) de cada classe representada pelos rótulos previamente definidos. Trata-se de uma rede neural convencional, onde todos os neurônios são ligados entre si, os quais possuem pesos e *bias* treináveis.

Para a classificação final do vetor de entrada é utilizado algum algoritmo de perda (*loss function*), por exemplo o *softmax* [Bridle 1990a] ou um SVM. O objetivo aqui pode ser a obtenção de um rótulo de classificação ou de uma distribuição de probabilidades para todos os rótulos possíveis, dependendo do tipo da rede e dos algoritmos empregados.

3. Técnica de *Word Embedding*

Conforme explicado nas seções anteriores as CNNs têm na camada de entrada de dados uma matriz de dimensões fixas, para que seja possível ser realizada a operação de convolução e para que se obtenha outras matrizes de tamanho fixo nas camadas

subsequentes. Em uma rede neural convolucional que classifica sentenças textuais isso significa uma matriz de duas dimensões, onde a primeira dimensão é a quantidade de palavras da sentença ou texto e a segunda dimensão é a quantidade de caracteres em cada palavra, conforme mostrado na Figura 2. Um problema, entretanto, é o fato de que é comum que dados como textos ou sentenças avulsas possuam quantidades variáveis de palavras. Além disso, outro problema que surge é que as palavras de uma linguagem qualquer possuem quantidades variáveis de caracteres, comprometendo assim o requerimento de matrizes de tamanho fixo em ambas as dimensões. Por fim, outra necessidade que surge é encontrar uma forma ideal para representar informações textuais em forma de matrizes numéricas de tamanho fixo, sem perder as relações semânticas, sintáticas e de contexto encontradas nas sentenças a serem fornecidas para a rede. Uma forma de contornar este problema é o uso de representações distribuídas de palavras, também conhecido como *word embedding*.

A técnica de *word embedding* consiste em capturar características sintáticas e semânticas de palavras extraídas de textos sem a intervenção humana ou processamento dependente de linguagem específica. As características capturadas pela técnica de *embedding* são independentes do tipo de tarefa a ser feito, o que as tornam ideais para modelagem de linguagem em geral [Chen et al. 2013].

A captura de características pode ser feita através de diversos modos, porém os métodos que vêm se destacando nos últimos anos e que têm sido utilizados na literatura recente são aqueles baseados em redes neurais. Estes modelos de sentenças podem variar desde o *bag-of-words* ou o *bag-of-n-grams* baseado em redes neurais até aqueles modelos que utilizam redes neurais recursivas ou de *time-delay* com convoluções. Os modelos neurais de sentenças para *word embedding* têm a vantagem de poderem ser treinados para obter vetores genéricos de tamanho fixo para as palavras ou sentenças, mantendo nesses vetores as características e relações semânticas, sintáticas e de contexto que as palavras e frases possuem entre si. Com isso pode se aplicar os vetores de palavras resultantes do treinamento no modelo de sentenças como entrada de uma rede neural convolucional, podendo prosseguir pelas camadas subsequentes da rede [Kalchbrenner et al. 2014].

4. Arquitetura do modelo proposto

O modelo de rede neural convolucional proposto neste artigo é inspirado no trabalho de Kim (2014). Este modelo apresenta uma complexidade baixa em relação a quantidade de camadas usadas ao longo da rede, mantendo resultados significativos para diversas tarefas de classificação de textos. Ao manter um número baixo de camadas e operações no modelo diminui-se também a complexidade computacional para que tais redes sejam treinadas, otimizando e acelerando o processo de validação do modelo para diversas combinações de hiper-parâmetros [Kim 2014 e Severyn e Moschitti 2015]. A Figura 3 mostra uma visualização de alto nível da arquitetura a ser utilizada para esta tarefa de classificação de sentimentos em texto.

Neste trabalho, o modelo proposto é composto por uma camada de convolução com uma função não-linear, uma camada de subamostragem e por fim um classificador *softmax*. Tais camadas e outros componentes essenciais serão explicados em maiores detalhes nas próximas subseções, iniciando pela entrada da rede composta pela matriz

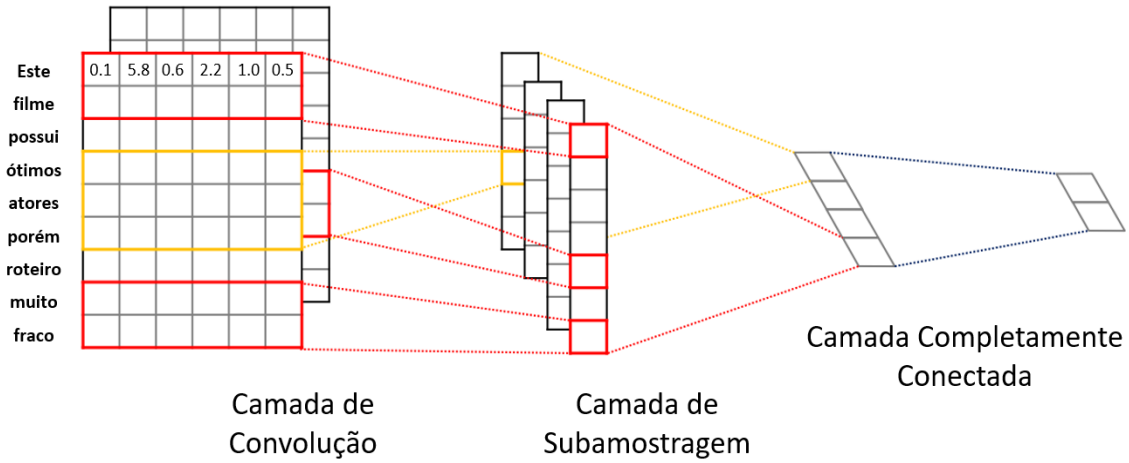

Figura 3. Modelo proposto para este trabalho. Adaptado de Kim (2014).

de sentenças, camada de convolução e os mapas de características, a camada de subamostragem, o método de regularização usado para treinamento e a camada de classificação com o uso da função *softmax*.

4.1. Matriz de Sentenças

A matriz de sentenças representa a entrada dos dados textuais deste modelo de rede neural convolucional. Ela é obtida a partir da técnica de *word embedding* detalhada na Seção 3. Trata-se de uma matriz *S*, de dimensões *d* por *s*, onde *d* representa o tamanho constante do vetor de *embedding* para cada palavra do texto, e *s* denota a quantidade de palavras por texto, cujo valor numérico também é constante.

Para este trabalho decidiu-se utilizar tanto vetores de *embedding* pré-treinados em bibliotecas de *word embedding*, tendo o *word2vec* [Mikolov et al. 2013] e o *GloVe* [Pennington et al. 2014] como exemplos, quanto o uso de *embeddings* inicializados randomicamente. Para os vetores pré-treinados será utilizado o repositório de *word embeddings* em Português do Brasil do NILC (Núcleo Interinstitucional de Linguística Computacional), mantido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP). Em contrapartida os vetores de *embedding* aleatórios serão inicializados no início do treinamento da rede com a ajuda das funções auxiliares de *embedding* de texto da biblioteca *TensorFlow* [Abadi et al. 2015].

4.2. Camada de Convolução

O objetivo da camada de convolução é extrair características e padrões comuns encontrados nas matrizes de sentenças usadas como entrada para o treinamento da rede. Para isso, vários filtros *F*, de tamanho *d* por *m* são percorridos pela matriz de sentença, realizando o produto escalar entre o filtro e o valor de entrada, a adição de um valor de *bias* e a aplicação de uma função de ativação (não-linearidade), conforme descrito anteriormente na Seção 2.1. Nesta operação, *d* representa o tamanho do vetor de *embedding* e *m* representa a largura da janela de observação do filtro, ou seja, a quantidade de palavras que serão processadas a cada mudança de posição do filtro durante a convolução. É importante notar que uma das dimensões dos filtros

corresponde ao tamanho do vetor de *embedding*, utilizando assim a totalidade das informações obtidas com a técnica de *word embedding*. A dimensão *m* pode ser modificada e aprimorada, ou mesmo ser usado mais de um valor de *m* para diferentes filtros. A função não-linear a ser aplicada nesta camada de convolução é a *ReLU* [Nair e Hinton 2010].

Após a aplicação de cada filtro sobre a matriz de sentenças serão obtidos os mapas de características, os quais contém informações codificadas de alto nível a respeito do texto ou sentença, de acordo com os pesos treinados em cada filtro. A partir daqui é possível utilizar os mapas de características em uma operação de *pooling*, conforme será descrito em seguida.

4.3. Camada de Subamostragem

Para o processamento na camada de subamostragem (*pooling*) é utilizado a técnica do *max-pooling*. Para cada mapa de características obtido na camada de convolução, uma janela de subamostragem é percorrida pelo vetor, extraindo apenas o maior valor observado naquela janela. Essa operação garante a redução das dimensões dos mapas de características, reduzindo a complexidade computacional. Além disso, conforme explicado na Seção 2.2, o *max-pooling* preserva as caraterísticas mais importantes do mapa de características, mesmo reduzindo as dimensões do vetor processado [Boureau et al. 2010].

4.4. Regularização

Um dos problemas observados em redes neurais Convolucionais e em outros métodos de aprendizagem profunda é o fenômeno da sobreposição (*overfitting*), especialmente quando se faz o uso de uma base de dados com um número insuficiente de amostras. Ao treinar uma rede neural qualquer com uma base de tamanho insuficiente, os parâmetros aprendidos durante o treinamento representam apenas um subconjunto pequeno de exemplos dos dados que se quer classificar com a rede. Isso significa que a rede irá se comportar muito bem durante o treinamento, porém irá falhar na maioria das vezes assim que forem apresentados exemplos fora do conjunto de treino ou teste, causando assim o *overfitting* [Srivastava et al. 2014].

Para contornar este problema, o modelo proposto aqui se utilizará da técnica de *dropout*. Através do *dropout* previne-se a possível co-adaptação de neurônios removendo temporariamente uma proporção *p* de neurônios de acordo com uma probabilidade pré-estabelecida durante o treinamento [Srivastava et al. 2014].

4.5. *Softmax*

Após obter os vetores de características e passá-los a camada de subamostragem, inicia-se o processo de classificação. Para tanto, uma camada completamente conectada de neurônios com o algoritmo de *softmax* recebe este vetor processado e calcula a distribuição de probabilidades sobre o rótulo e classes estabelecidos inicialmente. Por fim consegue-se extrair o rótulo que, segundo a rede, possui a maior probabilidade de representar a entrada fornecida a rede neural convolucional.

5. Experimentos sobre a base de dados

Definidos os conceitos básicos a respeito das redes neurais convolucionais e levando em conta a arquitetura descrita na Seção 4, nesta seção será explicado como é composta a base de dados (ou *dataset*) de entrada para a etapa de treinamento, os hiper-parâmetros escolhidos para cada uma das camadas da rede e o método de implementação e treinamento do modelo de classificação proposto.

5.1. Base de dados

O *dataset* a ser utilizado para treinamento da rede neural convolucional proposta neste trabalho é composto por um conjunto de textos extraídos da Internet. Os dados são artigos completos extraídos do site *BuzzFeed Brasil*, usando um *crawler* de páginas *web*. As informações extraídas dos artigos incluem o seu título, a descrição, o conteúdo do artigo e as reações dos leitores a respeito do conteúdo exposto. Esta base de dados é fruto do trabalho – ainda não publicado – de um grupo de pesquisadores do Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná (PUCPR).

Do conjunto de informações extraídas pode-se destacar dois grupos distintos: os dados textuais usados no treinamento da CNN e os dados de classificação de cada informação textual em rótulos ou classes distintas. O primeiro grupo são os dados textuais que representam o conjunto de valores da matriz de sentenças usada na primeira camada de convolução da rede. Estes dados são compostos pelo conteúdo do título, descrição e texto do artigo extraído. O segundo grupo são as reações ou sentimentos dos leitores em relação a cada artigo. No site do *BuzzFeed Brasil* os leitores, após finalizarem a leitura do artigo, têm a possibilidade de expressar a sua reação ao texto escolhendo um dos tipos de reação apresentados no fim da página. Estas reações estão separadas em oito classes distintas, representando diversas reações negativas, neutras ou positivas. Através do método de supervisão distante (*distant supervision learning*) [Mintz et al. 2009], assume-se como verdade absoluta (*ground truth*) o fato de que as reações ou sentimentos extraídos para cada texto representam o verdadeiro sentimento de tal texto, podendo serem apresentadas a rede neural convolucional como um vetor rótulos das matrizes de sentenças durante o treinamento.

Esta base de dados possui *8251* amostras de artigos juntamente com as devidas classificações de sentimentos rotuladas pelo método de *distant supervision learning*. Antes dos dados serem passados para a CNN é feito um processamento textual mínimo, onde ocorre remoção de caracteres inválidos, limpeza e concatenação dos dados textuais e preparação do vetor de rótulos com as reações dos textos.

5.2. Hiper-parâmetros

Além da definição da arquitetura do modelo, exposta na Seção 4, é necessário definir todos os hiper-parâmetros da CNN que farão com que a rede se comporte da melhor maneira possível nesta tarefa de classificação de sentimentos em textos. Embora a arquitetura seja de complexidade simples e de fácil implementação e treinamento, um ponto negativo do uso de redes neurais convolucionais é a dificuldade no momento da definição dos hiper-parâmetros ideais. Atualmente, não é claro o processo e a metodologia da estimação de hiper-parâmetros de CNNs devido a quantidade de

variáveis que podem ser alteradas e à sensibilidade da rede em relação a estas alterações. O espaço total de possíveis arquiteturas de modelos e configurações de hiper-parâmetros é muito vasto [Zhang e Wallace 2015]. Devido a este fato, os hiper-parâmetros usados para o modelo deste trabalho foram aqueles que apresentaram os melhores resultados nos diversos testes que foram feitos, embora seja encorajado a busca por configurações ainda melhores de hiper-parâmetros, resultando em uma precisão de classificação melhor que aquela apresentada neste trabalho.

O modelo proposto neste artigo necessita da configuração de, no mínimo, seis hiper-parâmetros globais:

- Definição da representação da matriz de sentenças de entrada: o vetor de entrada *V*, de duas dimensões, será de tamanho *d = 300*, representando o tamanho do *embedding* para cada palavra, e tamanho *s* representando a quantidade de palavras no texto ou sentença, definido por:

$$s = \lfloor \mu + 2 * \sigma \rfloor \qquad \text{[Eq. 4]}$$

  onde μ é a média aritmética da quantidade de palavras de todos os textos do *dataset* e σ representa o desvio padrão da quantidade de palavras nos textos. Quando um texto possui quantidade menor de palavras do que *s*, adiciona-se um *token* especial *s - n* vezes, onde *n* representa a quantidade de palavras do texto em questão. Quando um texto possui quantidade maior de palavras comparado com *s*, remove-se estas palavras adicionais até que se atinja o tamanho *s*.

- Tamanho da região dos filtros de convolução: será utilizado filtros *F* de convolução de duas dimensões, com tamanho *d = 300*, compatível com o tamanho do *embedding* definido na matriz de sentenças, e uma janela de palavras *m = [3, 4, 5]*, representando três tipos distintos de tamanho de filtro.

- Número de mapas de características: cada filtro de convolução *F* terá 128 mapas de características distintos. Isso resultará em *384* mapas de características para filtros de convolução de janelas *m = [3, 4, 5]*.

- Função de ativação: será usado como função de ativação o *Rectified Linear Units*, ou *ReLU*.

- Estratégia de subamostragem: para a operação de subamostragem será utilizado o *max-pooling*, com uma janela de duas dimensões de tamanho *(d - tamanho do filtro + 1)* por *1*, obtendo assim as características mais marcantes do mapa de características.

- Regularização: para evitar *overfitting* e promover melhor confiabilidade nos resultados será utilizado a técnica de *dropout* com probabilidade *p = 0.5*.

5.3. Implementação e treinamento

A implementação e treinamento desta rede neural convolucional é feita com a ajuda do *framework* de aprendizagem profunda *TensorFlow 1.3.0*, do Google. A visão geral do modelo construído para o *TensorFlow* neste trabalho é mostrada na Figura 4. O processo de treinamento da rede proposta é feito com os hiper-parâmetros estabelecidos na subseção anterior. Para tanto, o modelo utiliza-se do método de otimização estocástica

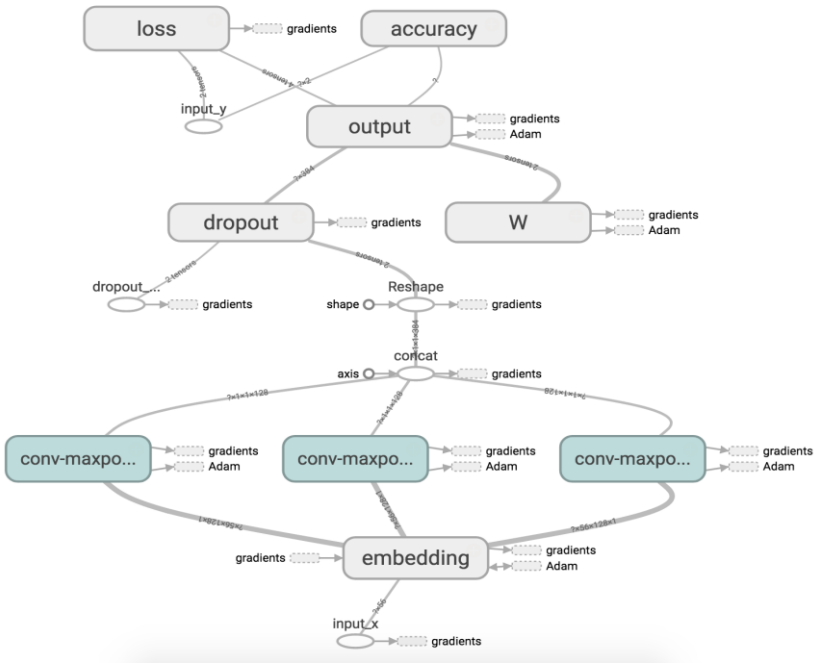

Figura 4. Visualização gráfica da arquitetura da CNN (retirado do *TensorFlow*).

Adam [Kingma e Ba 2014] para a otimização dos parâmetros e redução da função de perda. Mini-*batches* de *64* exemplos de dados de entrada são usados a cada passo de treinamento, sendo que a técnica de *early stopping* usada foi estabelecer um número fixo de *60 epochs* para treinamento devido a rápida convergência da estatística de perda. A cada *50* iterações de treinamento a acurácia e a perda da rede são avaliadas, e o melhor modelo sempre é mantido salvo para uso posterior. Nesse âmbito, *melhor modelo* significa aquele que possui menor perda na validação, e não a melhor acurácia geral. Tal decisão foi feita para se obter o modelo que possui a melhor generalização.

As instâncias da base de dados utilizada possuem oito classes distintas de sentimentos para os artigos em texto, sendo elas representadas por reações comumente utilizadas em redes sociais. Especificamente, as oito reações são: *cute* (fofo), *fail* (fracasso), *hate* (ódio), *lol* (engraçado), *love* (amor), *omg* (espanto), *win* (vitória) e *wtf* (indignação). Para este trabalho foi treinado um classificador para cada uma das reações, no fim obtendo-se oito classificadores distintos para serem combinados. No treinamento de cada classificador foi utilizado uma base de dados positiva contendo todas as instâncias positivas de tal reação, e uma base negativa com exemplos negativos da classe. Ao considerar o fato de que os usuários do *BuzzFeed Brasil* podem escolher qualquer sentimento em relação a determinado artigo, um artigo pode ser representado por mais de uma reação ao mesmo tempo, assumindo caráter multiclasse. Portanto algumas instâncias de textos podem aparecer em mais de um classificador no momento do treinamento e validação.

A base de dados de textos em português do Brasil a ser utilizada neste trabalho pode ser considerada inédita, não possuindo, consequentemente, divisão oficial para a separação dos dados usados no treinamento e na validação. Para a validação do modelo proposto foi utilizado a técnica de *holdout*, pois técnicas mais elaboradas, como exemplo a validação cruzada com n-*folds*, se tornaram proibitivas pela indisponibilidade

de hardware de alto desempenho, visto que tais técnicas exigem tempo na ordem de horas – ou até mesmo dias – de treinamento quando se utiliza *hardware* não especializado para a tarefa. Dada as devidas limitações, o *dataset* é divido da seguinte maneira para treinamento e validação:

- *60%* do *dataset* é utilizado como dados para treinamento da CNN, sendo divido durante as *epochs* de treinamento pelo tamanho pré-estabelecido para os mini-*batches*.
- *20%* dos dados para validação dos parâmetros treinados, sendo frequentemente usados para a avaliação do desempenho do treinamento e para *tweaking* dos hiper-parâmetros.
- *20%* restante dos dados como teste do modelo final treinado, sendo usados somente na última etapa da execução do modelo implementado.

O repositório com o código fonte do modelo proposto está disponível no GitHub pessoal do autor, através da URL <https://github.com/lasobrinho/Sentiment-Analysis-CNN-PTBR>. A GPU usada para treinamento foi uma Nvidia GeForce GTX 1080.

5.4. Resultados

Após o treinamento e teste de cada um dos oito classificadores para cada reação foi executado o teste do modelo final usando os *20%* de dados separados incialmente para essa tarefa de validação com a técnica de *holdout*. Duas sessões de treinamento foram realizadas, sendo a primeira usando vetores de *word embedding* inicializados aleatoriamente e a segunda sessão usando vetores pré-treinados. Os resultados da acurácia obtida para cada classificador são apresentados na Tabela 1, com dados tanto para *embeddings* randômicos quanto para pré-treinados.

Tabela 1. Resumo das acurácias obtidas por classificador para cada reação

Classificador para cada sentimento	Acurácia para embeddings randômicos (%)	Acurácia para embeddings pré-treinados (%)	Instâncias positivas da classe
cute (fofo)	67.90	68.07	2843
fail (fracasso)	59.67	59.98	4099
hate (ódio)	61.30	65.06	3320
lol (engraçado)	70.82	73.51	2427
love (amor)	67.51	70.03	793
omg (espanto)	57.69	57.75	3805
win (vitória)	55.62	58.80	3381
wtf (indignação)	58.54	60.05	3806
Acurácia final combinada por média aritmética	**62.38**	**64.16**	

6. Discussão dos resultados

Analisando os resultados da Tabela 1, percebe-se que a acurácia final obtida por média aritmética atingiu um valor aproximado de *62%* e *64%* de taxa de acerto para as instâncias apresentadas no teste final, para *embeddings* randômicos e pré-treinados, respectivamente. A base de dados usada neste trabalho é inédita, logo não se pode fazer comparações com o objetivo de compreender se tal resultado obtido é bom ou ruim, fazendo com que este experimento inicial seja a base das comparações e o estado da arte para os trabalhos futuros usando este mesmo conjunto de dados.

Comparando individualmente o desempenho de cada classificador para os sentimentos distintos disponíveis na base de dados percebe-se a falta de consistência nos resultados obtidos. Para *embeddings* pré-treinados, classes como a *love* e a *lol* obtiveram acurácias altas, de aproximadamente *70%* e *73%* respectivamente, ao passo que outras classes obtiveram desempenho abaixo do desejado, por exemplo *omg* e *win*, tendo aproximadamente *58%* de acurácia em ambas as classes. Tal falta de consistência nos resultados individuais pode ser explicado pelo fato das instâncias presentes no *dataset* poderem ser caracterizadas mais de uma classe ao mesmo tempo. Isso gera um compartilhamento virtual de instâncias entre os oito classificadores, fato que pode ser prejudicial no treinamento da rede para determinadas reações.

Um resultado interessante apresentado pelo experimento realizado é a diferença de acurácia obtida entre o treinamento utilizando *embeddings* aleatórios e o treinamento com vetores pré-treinados. Obteve-se um acréscimo de *1.78%* de acurácia quando se utilizou *embeddings* pré-treinados, conforme mostrado nos dados da Tabela 2. Tal resultado é significante ao considerar-se o fato de que o tempo de treinamento e o uso de recursos de *hardware* é o mesmo quando comparado a utilização de *embeddings* aleatórios. Isso prova que é possível obter um ganho real de acurácia de maneira barata apenas por utilizar tais representações pré-treinadas de palavras.

Tabela 2. Comparação da acurácia para os diferentes tipos de *embeddings*

Classificador para cada sentimento	Ganho de acurácia com embeddings pré-treinados vs aleatórios (%)
cute (fofo)	0.18
fail (fracasso)	0.31
hate (ódio)	3.76
lol (engraçado)	2.68
love (amor)	2.52
omg (espanto)	0.07
win (vitória)	3.18
wtf (indignação)	1.51
Ganho médio de acurácia	**1.78**

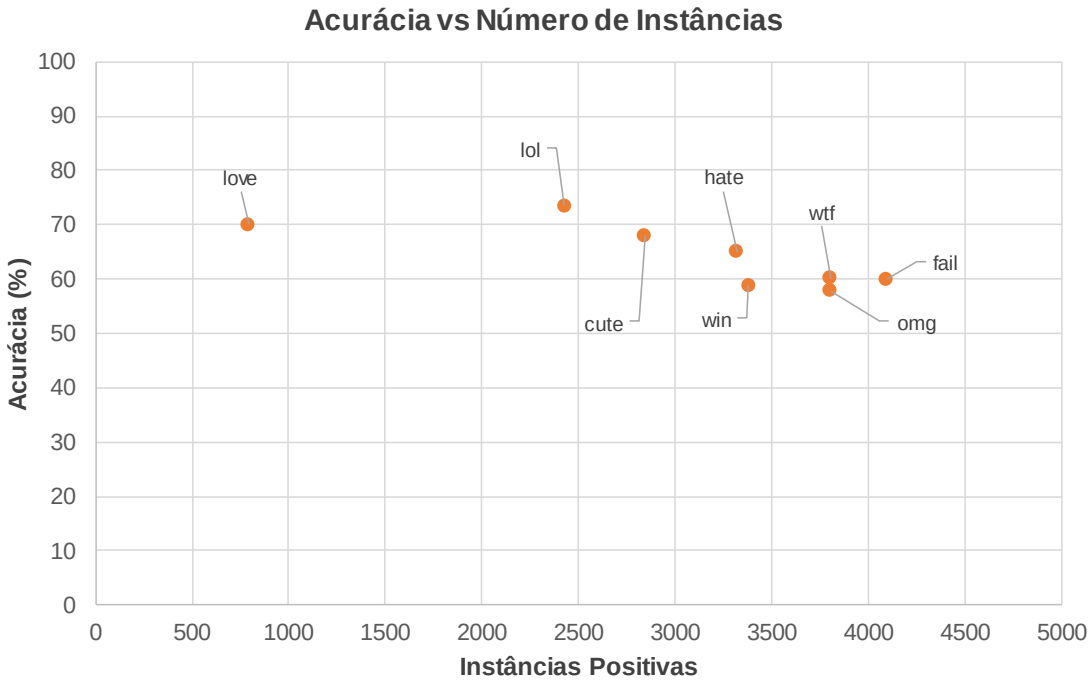

Figura 5. Gráfico de dispersão da acurácia dos classificadores e as instâncias

Para os classificadores treinados e validados não foi encontrado forte correlação entre o número de instâncias e a acurácia final obtida, conforme mostrado no gráfico de dispersão da Figura 5. Para *embeddings* pré-treinados, classes como a *love*, a qual possui somente *793* instâncias positivas, obteve uma performance satisfatória com precisão de aproximadamente *70%*, ao passo classes como a *omg*, que possui um número grande de instâncias, não foi capaz de obter uma boa performance. Pode-se concluir que para se obter uma acurácia alta é mais importante usar instâncias positivas mais específicas para tal classe de sentimento do que usar um número alto de instâncias de carácter generalista.

Um dos fatores para não se ter obtido resultados ainda melhores é o fato de que o tamanho da base de dados usada para este experimento é limitado. Conforme explicado anteriormente, Redes Neurais Convolucionais (e os demais métodos de aprendizagem profunda) são extremamente dependentes de *datasets* com dezenas de milhares (e talvez milhões) de instâncias. Aqui, neste experimento, usa-se uma base de aproximadamente *8000* instâncias, número indiscutivelmente baixo para o treinamento de CNNs.

Outro fator importante que pode comprometer os resultados deste experimento é a indisponibilidade de *hardware* dedicado para a tarefa de treinamento e validação de redes neurais convolucionais. Tal *hardware* é essencial para uma busca completa sobre os hiper-parâmetros ideais para a tarefa de classificação, algo que leva um tempo grande quando se usa *hardware* não dedicado. Como trabalhos futuros, sugere-se a obtenção de computadores ou *clusters* com componentes de alto nível para uma busca extensiva sobre a melhor combinação de hiper-parâmetros a fim de aumentar a acurácia dos classificadores.

7. Conclusão

Neste trabalho foi apresentado um método automatizado para a extração e classificação de sentimentos em textos na língua portuguesa do Brasil. Para tanto foi utilizado a técnica de Redes Neurais Convolucionais, com ênfase em processamento de linguagem natural. Foi utilizado uma base de dados com *8251* artigos textuais e reações dos leitores extraídos de um site de artigos textuais da rede pública de computadores, sendo que cada instância do *dataset* possui o texto completo do artigo e os respectivos sentimentos ou reações dos leitores, os quais pelo método de *distant supervision learning* são considerados como o verdadeiro sentimento ou reação para o respectivo texto através da aplicação do conceito de *ground truth*. A base possui oito diferentes sentimentos para os textos e, portanto, foram treinados oito classificadores CNN, um para cada sentimento.

Foram feitas duas sessões de treinamento, uma utilizando *embeddings* aleatórios e outra com *embeddings* pré-treinados. Dentre os classificadores de sentimentos em textos treinados nas duas sessões, as acurácias após treinamento e teste da rede se mostraram dentro do intervalo entre *55% e 74%*, com uma acurácia média final combinada de aproximadamente *62%* na predição de sentimentos usando *embeddings* aleatórios, e de aproximadamente *64%* quando usado *embeddings* pré-treinados. Como este é um experimento pioneiro sobre a base em questão não existem outros trabalhos para comparação do resultado, fazendo com que os resultados aqui presentes sejam a base comparativa e estado da arte para futuros experimentos neste mesmo *dataset*.

Para trabalhos futuros indica-se o uso uma base de dados com um número maior de instâncias e o uso de *hardware* dedicado somente a tarefa de treinamento e validação de Redes Neurais Convolucionais, fazendo com que seja possível, respectivamente, a melhoria da acurácia dos classificadores através do uso de representação de *embeddings* mais poderosos e a busca extensiva pelos melhores hiper-parâmetros do modelo de CNN sendo usado para a classificação de sentimentos em texto.

Referências

Yi, Jeonghee, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *IEEE Intl. Conf. on Data Mining (ICDM)*.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Second International Conference on Knowledge Capture*, Florida, USA.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: a survey. *Ain Shams Eng J* 5(4):1093–1113.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882v2*.

Bridle JS. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. 1989. In: *Fogelman-Soulié F, Hérault J (eds), Neurocomputing − Algorithms, Architectures and Applications. NATO ASI Series F68, Springer-Verlag, Berlin (D)*, pp 227-236.

Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *G.B. Orr and K.-R. Müller, editors, Neural Networks: Tricks of the Trade*, pages 9–50. Springer.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. 2015. Recent advances in convolutional neural networks, *arXiv preprint arXiv:1512.07108*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*.

Tao Wang, David J. Wu, Adam Coates, and Andrew Y Ng. 2012. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on Pattern Recognition*, pp. 3304–3308. IEEE.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010b. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Santiago, Chile, August 9-13, 2015, pages 959–962.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv:1301.3226*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent

Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pages 265–283.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification Using Machine Learning Techniques. In *Proceedings of EMNLP 2002*, pages 79-86.