UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM BIG DATA, DATA SCIENCE E DATA ANALYTICS

Roberto Martins da Silva

CLASSIFICAÇÃO DE IMAGENS: UM MÉTODO DE OTIMIZAÇÃO DE HIPERPARÂMETROS DE CONFIGURAÇÃO DE REDES NEURAIS CONVOLUCIONAIS

São Leopoldo
2019

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM BIG DATA, DATA SCIENCE E DATA ANALYTICS

Roberto Martins da Silva

CLASSIFICAÇÃO DE IMAGENS: UM MÉTODO DE OTIMIZAÇÃO DE HIPERPARÂMETROS DE CONFIGURAÇÃO DE REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Especialista em *Big Data, Data Science e Data Analytics*, pelo curso de Pós-Graduação Lato Sensu em *Big Data, Data Science e Data Analytics* da Universidade do Vale do Rio dos Sinos – UNISINOS.

Orientadora: Profa. Ms. Denise Bandeira Da Silva.

São Leopoldo
2019

Classificação De Imagens: Um Método De Otimização De Hiperparâmetros De Configuração De Redes Neurais Convolucionais

Roberto Martins da Silva

Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, Polo de Bento Gonçalves – 95700-000 – Bento Gonçalves, RS – Brasil

robertomsilva@edu.unisinos.br

Resumo. *A construção de um modelo de Rede Neural Artificial para reconhecimento de imagens exige um alto grau de especialização e simulações para se conseguir configurar todos os parâmetros necessários para um bom desempenho do modelo. Este artigo apresenta um método de análise dos principais hiperparâmetros de configuração de uma Rede Neural Convolucional de reconhecimento de imagens de forma a tornar mais acessível a tarefa de escolha dos parâmetros de configuração e que possa ser estendido a outros modelos de redes neurais que utilizam os mesmos parâmetros, encurtando assim o trabalho de construção do modelo. Passando pelos principais parâmetros de configuração, à medida em que vão sendo definidas as faixas ideais de valores, o método vai conduzindo aos parâmetros seguintes, permitindo o aprimoramento do resultado da rede neural e a experimentação com os parâmetros das etapas anteriores até se chegar ao resultado ideal. O processo se mostrou eficaz, uma vez que, partindo de um modelo já testado, foram removidas as otimizações e aplicados os testes do método, chegando a uma configuração final mais otimizada e com melhor desempenho que o modelo original.*

Abstract. *The construction of an Artificial Neural Network model for image recognition requires a high degree of specialization and simulations to be able to configure all the parameters necessary for a good model performance. This paper presents a method for analyzing the main configuration hyperparameters of a Convolutional Image Recognition Neural Network to make the task of choosing configuration parameters more accessible and can be extended to other neural network models that use them. parameters, thus shortening the model construction work. Going through the main configuration parameters, as the ideal ranges of values are defined, the method leads to the following parameters, allowing the improvement of the neural network result and experimentation with the parameters of the previous steps until the ideal result is reached. The process proved to be effective, since, starting from a previously tested model, the optimizations were removed and the method tests were applied, reaching a more optimized and better performing final configuration than the original model.*

1 Introdução

A visão computacional, que é a extração de informações de imagens e sequências de imagens, além de ser importante na interação humano-computador, pode ser usada por robôs para extrair informações de seu ambiente (BEHNKE, 2003). Técnicas de visão computacional também são usadas para a análise de imagens estáticas. A parte mais problemática da visão computacional é a interpretação das imagens capturadas. Este

problema tem dois aspectos principais: velocidade e qualidade de interpretação (BEHNKE, 2003).

Dentre as diversas técnicas de aprendizado de máquina, uma que tem se destacado de forma bastante acentuada é a de Redes Neurais Artificiais (BISHOP, 2006), que são baseadas no funcionamento dos neurônios de um cérebro biológico (HAYKIN, 2008). Quanto ao reconhecimento de imagens, as Redes Neurais Convolucionais, também baseadas em um modelo biológico, desta vez no córtex visual, permitem que se aplique filtros (*kernels*) sobre as imagens, extraindo padrões de características, ao invés de apenas analisar as cores dos pixels, o que dá um ganho significativo para se atingir um grau satisfatório de generalização, o que torna o modelo capaz de classificar as imagens recebidas mesmo que nunca as tenha analisado antes (LECUN et al, 1998).

Atualmente, existe uma infinidade de modelos de Redes Neurais Artificiais disponibilizadas em sites especializados (KERAS, 2019), (TENSORFLOW, 2019) e que permitem que se realize a tarefa de reconhecimento de imagens sem a necessidade de iniciar do zero a construção de um modelo. A construção de uma rede neural não é uma tarefa simples, nem possui uma regra pronta. Deve ser adaptada e configurada de acordo com os dados que se deseja analisar, pois cada conjunto de dados possui suas características e complexidades específicas. É necessário que se ajuste diversos parâmetros a fim de adequar a configuração da rede neural às imagens que serão processadas para que possa haver uma medida aceitável de acertos por parte do modelo e de acordo com a aplicação para a qual se destina o resultado da análise.

O ajuste dos parâmetros de aprendizado e dos hiperparâmetros do modelo geralmente é uma obra de engenharia que requer experiência especializada, regras práticas ou, às vezes, busca por força bruta (SNOEK, 2012).

Devido à importância da tarefa de reconhecimento de imagens de forma automática na área computacional e da grande dificuldade de se encontrar os valores ideais para configuração dos hiperparâmetros para se atingir um nível satisfatório de identificação dessas imagens, surge a questão: quais os principais parâmetros que interferem no desempenho da rede neural para reconhecimento de imagens e qual a melhor forma de se encontrar a combinação ideal desses parâmetros que permita atingir o melhor nível de desempenho nessa rede neural?

Dada a questão acima, o objetivo do presente artigo é definir uma sequência de etapas que ajudem a escolher as configurações dos principais parâmetros melhorando o desempenho da rede à medida em que forem sendo ajustados e que possa ser utilizada também para outros modelos de classificação que utilizem redes neurais convolucionais para classificação de imagens.

As seguintes etapas foram traçadas para que se consiga evoluir a partir dos dados básicos e se atingir o objetivo proposto:

- Selecionar um modelo padrão de Rede Neural Convolucional para classificação de imagens coloridas, removendo todas as otimizações de desempenho possíveis, a fim de ter um modelo básico padrão.
- Selecionar os principais parâmetros que interferem no desempenho do modelo.
- Testar e analisar as métricas de desempenho do modelo com diversos valores de cada um dos parâmetros selecionados a fim de encontrar a melhor

combinação de cada parâmetro que permita melhorar o desempenho do modelo, aumentando sua acurácia e reduzindo o *overfitting*.

- Comparar com os resultados obtidos durante o trabalho de pesquisa com o resultado original do modelo de rede utilizado a fim de demonstrar as melhorias obtidas com os ajustes de parâmetros.

Nas próximas seções, são apresentados a **Fundamentação Teórica** contendo os **Conceitos Básicos** de Aprendizado De Máquina (*deep learning*), direcionando para as Redes Neurais Artificiais com seus principais componentes e, mais especificamente, as Redes Neurais Convolucionais, objeto deste estudo, bem como as principais métricas utilizadas para apuração de desempenho, que serão importantes para a análise de resultados no final do projeto. Nessa mesma sessão são apresentados os **Trabalhos Relacionados** para os quais procurou-se um estabelecer um paralelo para que se pudesse identificar as lacunas e estreitar o foco do projeto, fornecendo dados novos sobre a otimização de parâmetros de Redes Neurais Artificiais. Em seguida descreve-se a **Metodologia de Pesquisa** utilizada, a **Definição do Método de Otimização** propriamente dito, iniciando com a descrição da base de dados e das ferramentas utilizadas e, posteriormente, a descrição do método proposto, acompanhada dos resultados das análises realizadas sobre a base de dados. Por fim, é apresentada uma **Discussão dos Resultados** onde são analisadas as decisões tomadas durante a aplicação do método e suas consequências nos resultados obtidos, bem como a **Conclusão** com propostas de trabalhos futuros que podem surgir a partir do presente artigo.

2 Fundamentação Teórica

Nesta sessão são apresentados os conceitos básicos relacionados ao tema de Classificação de Imagens com Redes Neurais e trabalhos relacionados com as respectivas características e as diferenças que justificam o artigo proposto.

2.1 Conceitos Básicos

São descritos os conceitos básicos de Redes Neurais, sobre os quais é definido o método proposto.

2.1.1 Aprendizado De Máquina

Segundo FLACH (2012), o aprendizado de máquina é o estudo sistemático de algoritmos e sistemas que melhoram seu conhecimento ou desempenho com experiência. Consiste de um programa de computador que memoriza a relação de dados de entrada com a respectiva classificação de saída permitindo que se faça generalizações (BISHOP, 2006).

Fundamentalmente, o aprendizado de máquina usa algoritmos para extrair informações de dados brutos e representá-las em algum tipo de modelo. Posteriormente este modelo é utilizado para inferir coisas sobre outros dados que ainda não foram modelados (PATTERSON e GIBSON, 2017).

O aprendizado de máquina se divide em Aprendizado Supervisionado e Aprendizado Não Supervisionado. As aplicações em que os dados de treino compreendem exemplos dos vectores de entrada, juntamente com os seus vectores alvo correspondentes, são conhecidos como problemas de aprendizado supervisionado. Casos como o exemplo de reconhecimento de imagens que será apresentado no decorrer desse artigo, em que o objetivo é atribuir cada vetor de entrada a um de um número finito de categorias distintas, são chamados de problemas de classificação.

2.1.2 Redes Neurais Artificiais

As redes neurais artificiais (RNA), comumente referidas como “redes neurais”, tem sido estudadas desde o seu início pelo reconhecimento de que o cérebro humano calcula de uma forma totalmente diferente do computador convencional. O cérebro é altamente complexo, não linear e de funcionamento paralelo no processamento de informações (HAYKIN, 2008).

As redes neurais são modelos matemáticos que possuem sua inspiração no funcionamento do cérebro e nas teorias psicológicas de aprendizado, especialmente a “Teoria do Aprendizado Neural” de Hebb (1949) (LUGER, 2013). Tais modelos têm seu uso amplamente difundido no campo da computação inteligente devido sua capacidade de aprendizagem e um melhor desempenho na classificação de dados altamente complexos (BISHOP, 2006).

2.1.3 Neurônio Artificial

A Rede Neural artificial é composta por “neurônios artificiais”, também chamados de Unidades de Processamento (HAYKIN, 2008). Na Figura 1 pode ser vista a estrutura básica de um neurônio artificial.

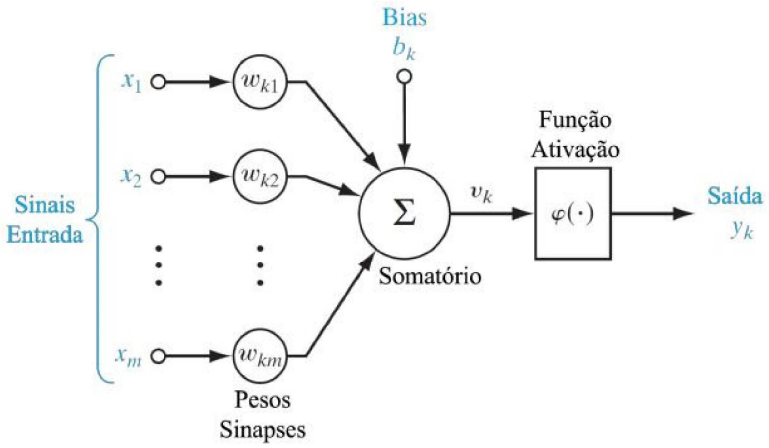

Figura 1 - Neurônio artificial

Na Figura 1 podem ser identificados os três elementos básicos de um neurônio:

1. Um conjunto de sinais de entrada, ou links de conexão, cada um dos quais é caracterizado por um peso próprio. Especificamente, um sinal *xj* na entrada *j* do neurônio *k* é multiplicado pelo peso *wkj*. O peso *w* em um neurônio artificial pode estar em uma faixa que inclui tanto valores negativos quanto positivos.
2. Um somatório para somar os sinais de entrada, ponderados pelos respectivos pesos do neurônio. Essas operações constituem um combinador linear.
3. Uma função de ativação para limitar a amplitude da saída de um neurônio. A função de ativação é também referida como uma função de limitação, na medida em que atinge (limita) o intervalo de amplitude permissível do sinal de saída para algum valor finito. Normalmente, a faixa de amplitude normalizada da saída de um neurônio é escrita como o intervalo unitário fechado [0,1] ou, alternativamente, [-1,1]. O modelo neural também inclui um bias aplicado externamente, denotado por *bk*. O bias *bk* tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se

é positiva ou negativa, respectivamente. Em termos matemáticos, pode-se descrever o neurônio *k* escrevendo as duas seguintes equações:

Equação 1:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

Equação 2:

$$y_k = \varphi(u_k + b_k)$$

onde *x1, x2, ..., xm* são os sinais de entrada; *wk1, wk2, ..., wkm* são os respectivos pesos sinápticos do neurônio *k*; *uk* é a saída do combinador linear devido aos sinais de entrada; *bk* é o bias; φ(·) É a função de ativação; e *yk* é o sinal de saída do neurônio.

2.1.4 Funções De Ativação

A função de ativação, como já citado anteriormente, tem o objetivo de atribuir um caráter não linear ao modelo de rede neural, o que, na prática, determina se o neurônio irá propagar o sinal recebido na entrada para os outros neurônios através da saída.

De acordo com NWANKPA et al (2018) dentre os principais tipos de função de ativação temos:

- Sigmoide: é uma função de ativação usada principalmente em redes neurais *feedforward*. As principais vantagens das funções sigmoides são fáceis de entender e são usadas principalmente em redes rasas. No entanto, a função sigmoide apresenta algumas desvantagens que incluem desaparecimento do gradiente, saturação de gradiente, convergência lenta e saída centrada diferente de zero, fazendo com que as atualizações de gradiente se propaguem em diferentes direções.

Outras formas de funções de ativação, incluindo a função de tangente hiperbólica, foram propostas para remediar algumas dessas desvantagens sofridas pela função sigmoide.

- Tangente Hiperbólica: tornou-se a função preferida em comparação com a função sigmoide, pois proporciona um melhor desempenho de treinamento para redes neurais multicamadas. No entanto, não conseguiu resolver o problema de desaparecimento do gradiente sofrido pelas funções sigmoides. A principal vantagem proporcionada pela função é que ela produz saída centrada em zero ajudando, assim, no processo de retropropagação.

As limitações da função tangente hiperbólica e sigmoide estimularam pesquisas adicionais em funções de ativação para resolver o problema e geraram a função de ativação ReLU.

- ReLU (Rectified Linear Unit): proposta por NAIR e HINTON (2010) e, desde então, tem sido a função de ativação mais amplamente utilizada para aplicações de *deep learning* com os melhores resultados até o momento. O ReLU é uma função de ativação

de aprendizado mais rápido. Oferece o melhor desempenho e generalização em *deep learning* em comparação com as funções de ativação sigmoide e tangente hiperbólica.

Essa função retifica os valores das entradas menores que zero, forçando-os a zero e eliminando o problema do desaparecimento do gradiente observado nos tipos anteriores de função de ativação. A função ReLU é bastante encontrada nas aplicações de classificação de objetos e reconhecimento de fala.

A principal vantagem de usar a função de ativação ReLU na computação é que ela garante uma computação mais rápida, uma vez que ela não calcula exponenciais e divisões. Outra propriedade da ReLU é que ela introduz esparsidade nas unidades ocultas à medida que esmaga os valores entre zero e máximo. No entanto, a ReLU tem uma limitação facilmente superável em comparação com a função sigmoide que é o *overfitting*, mas com o uso da técnica de *dropout* é possível reduzir o efeito do *overfitting*, melhorando o desempenho de redes neurais profundas.

A função de ativação ReLU e tem sido utilizada em diferentes arquiteturas de *deep learning*, dentre elas as arquiteturas de redes neurais convolucionais.

Na Figura 2 pode-se visualizar o gráfico de cada uma das funções de ativação apresentadas acima.

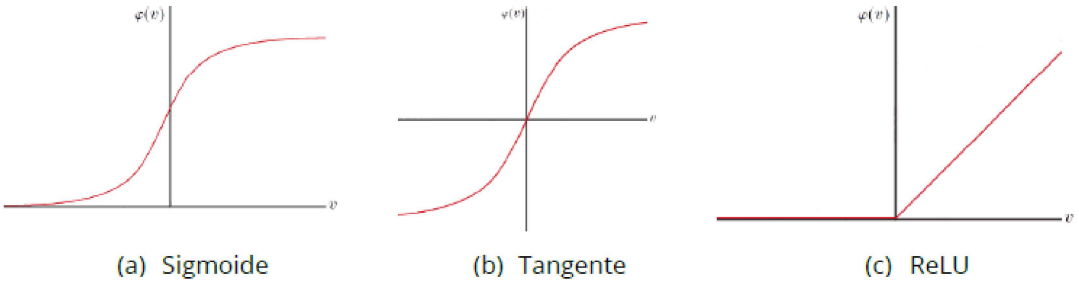

Figura 2 - Representação gráfica das funções de ativação

Segundo BISHOP (2006), há uma escolha natural da função de ativação da unidade de saída e da função de erro correspondente, de acordo com o tipo de problema a ser resolvido. Para regressão usamos saídas lineares e um erro de soma dos quadrados, para classificações binárias (múltiplas independentes) usamos saídas sigmoides logísticas e uma função de erro de entropia cruzada, e para classificação multiclasse usamos saídas *softmax* com a entropia cruzada multiclasse correspondente a função de erro. Para problemas de classificação envolvendo duas classes, podemos usar uma única saída sigmoide logística ou, alternativamente, podemos usar uma rede com duas saídas com uma função de ativação de saída *softmax*.

2.1.5 Estrutura Das Redes Neurais Artificiais

Uma rede neural é formada pelo conjunto de unidades de processamento (neurônios) organizadas em camadas, como pode ser visto na Figura 3.

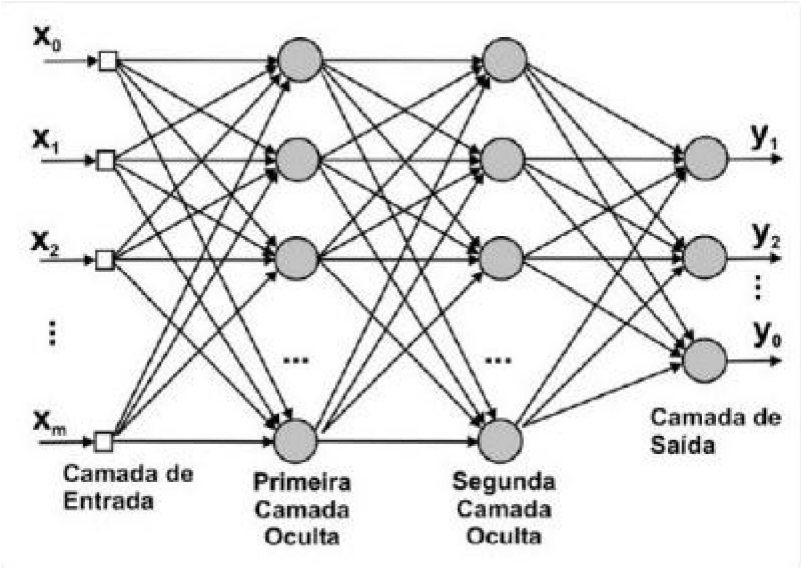

Figura 3 - Organização das camadas da rede neural artificial

A primeira e a última camada são chamadas de camada de Entrada e de Saída, respectivamente. As demais camadas intermediárias, as camadas ocultas, são assim chamadas porque não se tem acesso aos valores recebidos e gerados por elas. Cada nó nas camadas corresponde a uma unidade de processamento, que calcula sua saída e alimenta a entrada das camadas seguintes.

2.1.6 *Backpropagation*

De modo a evitar altas taxas de erro e possibilitar o aprendizado da rede através do ajuste de seus pesos, um dos algoritmos utilizados é a retropropagação (*backpropagation*, em inglês). A abordagem adotada consiste em iniciar na camada de saída e propagar o erro retroativamente através das camadas ocultas. O algoritmo *backpropagation* tem como principal objetivo a minimização da função de erro, atribuindo aos neurônios uma parcela de culpa pela diferença na saída e ajustando os pesos de forma correspondente (HAYKN, 2008).

2.1.7 Descida do Gradiente

A Descida do Gradiente (do inglês, *gradient descent*) é um algoritmo genérico de otimização capaz de encontrar e ajustar os parâmetros de forma iterativa, a fim de minimizar uma função de custo. O algoritmo mede a mudança em todos os pesos da rede com relação a taxa de erro dos parâmetros, sendo visto como a taxa de variação da função de custo. O gradiente ainda pode ser considerado como uma derivada parcial em relação às suas entradas. Portanto, quanto maior o gradiente, mais íngreme a inclinação e mais rápido o modelo pode aprender (GOODFELLOW et al, 2016).

O funcionamento do gradiente é condicionado a dois parâmetros principais: a **taxa de aprendizagem** e a **função de custo**. O primeiro, é bastante conhecido nas redes neurais e se refere a margem de alteração nos parâmetros. Já o segundo, é conhecido por tratar-se da função utilizada para mensurar o desempenho do algoritmo.

O cálculo do gradiente ocorre para cada variável até que seja encontrado o valor ideal para esta, comparando-a com o passo anterior e o futuro. Tal caminhada é conhecida como a busca pelo mínimo local ou global. O mínimo local é um vale localizado na função

em que o valor do custo é minimizado, porém não é o menor valor da função para a variável. Logo, o objetivo do gradiente será encontrar o mínimo global da função de custo para cada variável.

A taxa de aprendizagem é capaz de regular o aprendizado e a caminhada do gradiente na busca pelo mínimo global. Na Figura 4 é demonstrado o impacto da variação entre taxas de aprendizagem grandes e pequenas no processo de avaliação dos parâmetros. No primeiro caso, o gradiente faz caminhadas mais longas, pode encontrar o mínimo global mais rápido e pode evitar mínimos locais com mais facilidade, porém é possível que nunca encontre o mínimo global. Enquanto isso, a taxa de aprendizagem menor tem maiores chances de encontrar o mínimo global, porém levará mais tempo na execução do gradiente e existe a possibilidade de estagnar em mínimos locais. Portanto, deve-se buscar uma taxa de aprendizado equilibrada a fim de não comprometer o resultado da busca do mínimo global (BISHOP, 2006).

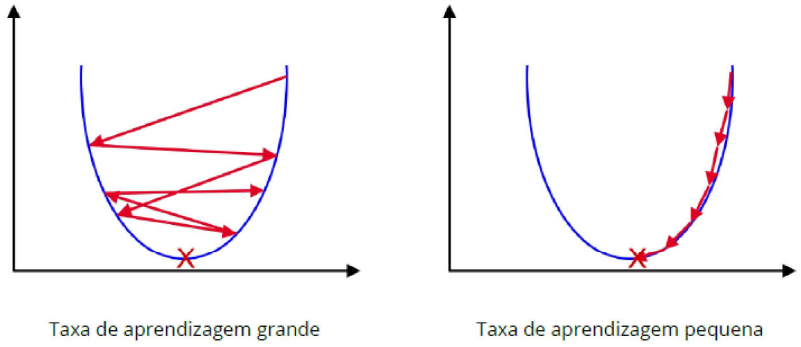

Figura 4 - Representação gráfica das taxas de aprendizagem

2.1.8 Mini Lotes (*Mini-Batch*)

De acordo com RUDER (2016), existem três variantes do *gradiente descent*, que diferem na quantidade de dados que usamos para calcular o gradiente da função de custo. Dependendo da quantidade de dados, fazemos uma troca entre a precisão da atualização do parâmetro e o tempo necessário para executar uma atualização.
A primeira das variantes é a descida do gradiente em lote (*Batch Gradient Descent*). Esta calcula o gradiente da função de custo para todo o conjunto de dados de treinamento, podendo ser muito lenta e impraticável para conjuntos de dados com tamanho maior do que a memória disponível.

A segunda variante é conhecida como descida do gradiente estocástico (*Stocastic Gradient Descent*). Esta, por sua vez, executa uma atualização de parâmetro para cada exemplo de treinamento. Se a descida do gradiente em lote realiza cálculos redundantes para grandes conjuntos de dados, pois recalcula gradientes para exemplos semelhantes antes de cada atualização de parâmetro, a descida do gradiente estocástico elimina essa redundância executando uma atualização por vez. Portanto, geralmente é muito mais rápido, e permite saltar para mínimos locais novos e potencialmente melhores. Por outro lado, isso acaba complicando a convergência ao mínimo exato. Entretanto, quando diminuímos lentamente a taxa de aprendizado, a descida do gradiente estocástico mostra o mesmo comportamento de convergência que a descida do gradiente em lote, quase certamente convergindo para o mínimo local ou global desejado.

Por fim, a terceira variante é a Descida de Gradiente de Mini lote (*Mini-batch Gradient Descent*). Esta, tira o melhor dos dois mundos e executa uma atualização para cada mini lote de *n* exemplos de treinamento. Dessa forma, reduz a variação das atualizações de parâmetros, o que pode levar a uma convergência mais estável.

Os tamanhos comuns de mini lotes variam entre 50 e 256, mas podem variar para diferentes aplicações. A descida do gradiente de mini lote é o algoritmo de escolha no treinamento de uma rede neural. Também é ideal que se opte por tamanhos de mini lotes múltiplos de 2, como 32, 64, 128, 256 e assim por diante.

Outro aspecto importante a considerar com relação ao uso de mini lotes, refere-se à relação entre o tamanho do mini lote e a taxa de aprendizado. Conforme apresentado por SMITH et al (2018), na tentativa de aumentar o tamanho do passo e reduzir o número de atualizações de parâmetros necessários para treinar um modelo, passou-se a utilizar lotes maiores. Infelizmente, quando aumentamos o tamanho do lote, a precisão do conjunto de testes geralmente cai. Naquele estudo, é proposta um tamanho ideal de lote proporcional à taxa de aprendizado. Quando se reduz a taxa de aprendizado, esta sofre uma deterioração. Nesse caso, propõe-se, ao invés de reduzir a taxa de aprendizado, que se aumente o tamanho do lote. Essa estratégia atinge um desempenho de modelo quase idêntico no conjunto de testes com o mesmo número de épocas de treinamento, mas com um número significativamente menor de atualizações de parâmetros.

2.1.9 *Deep Learning*

Deep learning corresponde à área de aprendizado de máquina que permite que modelos computacionais compostos de múltiplas camadas de processamento aprendam representações de dados com múltiplos níveis de abstração. Esses métodos melhoraram drasticamente o estado da arte em reconhecimento de fala, reconhecimento de objetos visuais, detecção de objetos e muitos outros domínios, como na área de saúde. *Deep learning* desmembra uma estrutura complexa em grandes conjuntos de dados usando o algoritmo de retropropagação para indicar como a rede deve alterar seus parâmetros internos que são usados para calcular a representação em cada camada da representação na camada anterior (YANN et al, 2015).

Segundo GOODFELLOW et al (2016), *deep learning* consiste em permitir que os computadores aprendam com a experiência e compreendam o mundo em termos de uma hierarquia de conceitos, com cada conceito definido em termos de sua relação com conceitos mais simples. Esta estrutura de conceitos se utiliza de modelos de redes neurais artificiais, que são baseadas na estrutura de funcionamento do cérebro humano.

Deep learnig tornou-se possível a partir da evolução das redes neurais, apresentando as seguintes características (PATTERSON e GIBSON, 2017):

- Mais neurônios do que redes anteriores;
- Maneiras mais complexas de conectar camadas / neurônios em redes neurais;
- Explosão na quantidade de energia computacional disponível para treinar;
- Extração automática de recursos.

2.1.10 Redes Neurais Convolucionais (CNN)

Uma rede convolucional é um neurônio artificial multicamadas, criado especificamente para reconhecer formas bidimensionais com alto grau de invariância à tradução, escala, distorção e outras formas de distorção. Essa tarefa é aprendida de maneira supervisionada por meio de uma rede cuja estrutura inclui as seguintes características (LECUN et al, 1998):

a) Extração de recursos: cada neurônio obtém suas entradas de um campo receptivo local na camada anterior, forçando-o a extrair recursos locais. Uma vez que um recurso tenha sido extraído, sua localização exata se torna menos importante, desde que sua posição relativa a outros recursos seja preservada aproximadamente.

b) Mapeamento de recursos: cada camada computacional da rede é composta de múltiplos mapas de características (filtros, ou *kernels*), com cada mapa de características sendo na forma de um plano dentro do qual os neurônios individuais compartilham o mesmo conjunto de pesos sinápticos. Esta etapa tem os seguintes efeitos:

• invariância de deslocamento: forçada na operação de um mapa de características através do uso de convolução com um *kernel* de tamanho pequeno, seguido por uma função sigmoide;

• redução do número de parâmetros livres, realizados através do uso de compartilhamento de peso.

c) Subamostragem: cada camada convolucional é seguida por uma camada computacional que executa a média local e a subamostragem, em que a resolução do mapa de recursos é reduzida. Essa operação tem o efeito de reduzir a sensibilidade da saída do mapa de recursos para mudanças e outras formas de distorção.

Um modelo de rede convolucional é apresentado na Figura 5, com suas principais camadas e funções.

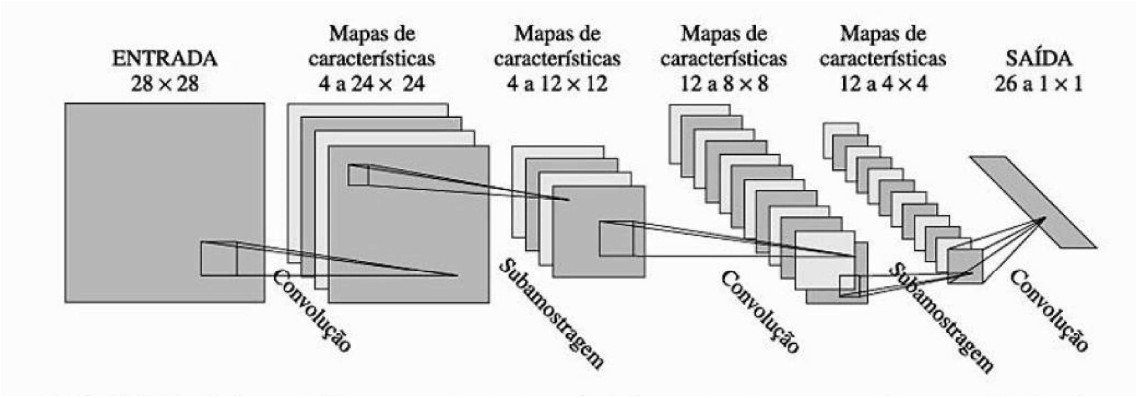

Figura 5 - Modelo de rede neural convolucional

As camadas de convolução podem aplicar diversos filtros sobre uma única imagem, cada qual captando um sinal diferente. É possível imaginar as camadas iniciais como filtros por linhas horizontais, verticais e diagonais no sentido de criar um mapa das bordas na imagem. As redes utilizam tais filtros para obter vetores de características (também conhecidos como mapa de recursos ou *feature maps*). Portanto, a rede pode possuir *n* camadas subjacentes de convoluções. O mesmo vale para os agrupamentos (*pooling*) até que seja realizada a transformação (*flatten*) e posterior classificação.

Os filtros possuem como objetivo encontrar as principais características da imagem. Para aplicá-los, uma série de parâmetros são necessários. Dentre eles, temos:

- *Input size* – O tamanho em pixels aceitos como entrada da rede;

- *Filter* – O tamanho dos filtros aplicados;

- *Padding* – Utilizado para forçar a preservação do tamanho da imagem na saída das convoluções através de preenchimento com valores zeros nas bordas;

- *Stride* – A quantidade de pixels que a janela dos filtros deve realizar o deslocamento por vez.

2.1.11 *Dropout*

Redes neurais profundas, com um grande número de parâmetros, são sistemas de aprendizado de máquina muito poderosos. No entanto, o *overfitting* é um problema sério em tais redes. O *dropout* é uma técnica para resolver esse problema. A ideia-chave é descartar aleatoriamente unidades (juntamente com suas conexões) da rede neural durante o treinamento. Isso evita que as unidades se adaptem demais aos dados reduzindo o *overfitting* e dando grandes melhorias em relação a outros métodos de regularização. O *dropout* melhora o desempenho de redes neurais em tarefas de aprendizado supervisionadas em visão, reconhecimento de fala, classificação de documentos e biologia computacional (SRIVASTAVA et al, 2014).

A Figura 6 demonstra o efeito da aplicação do *dropout* nas Unidades de Processamento.

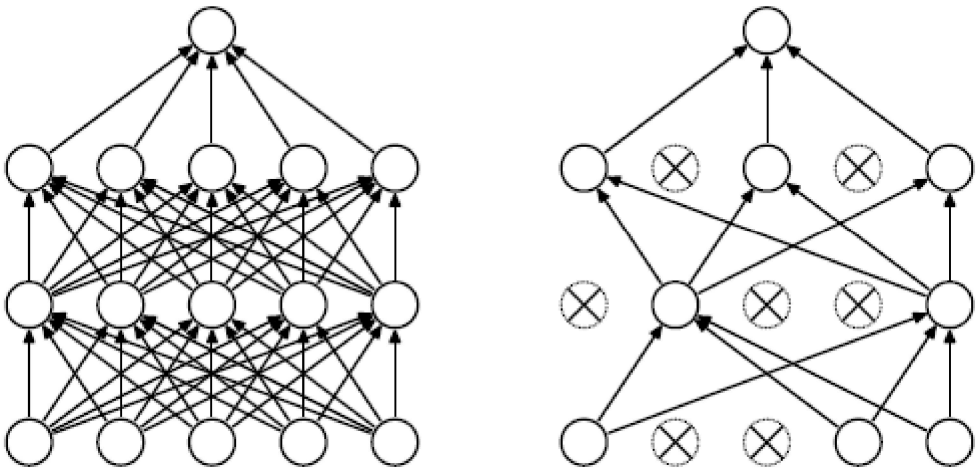

Figura 6 - Aplicação do *dropout* nas unidades de processamento

2.1.12 Métricas De Desempenho

Para avaliação dos resultados, existem quatro diferentes métricas primárias que são utilizadas como base para extrair estatísticas de desempenho dos modelos:

- Verdadeiros positivos ou *True Positives* (TP): são instancias positivas que foram classificadas como positivas;

- Falsos positivos ou *False Positives* (FP): são instancias negativas que foram incorretamente classificadas como positivas;

- Verdadeiro negativos ou *True Negatives* (TN): são instancias negativas que foram classificadas corretamente como negativas;

- Falsos negativos ou *False Negatives* (FN): são instancias positivas que foram incorretamente classificadas como negativas.

Essas métricas são utilizadas para elaborar a Matriz de Confusão dos resultados esperados e preditos.

A partir da matriz de confusão é possível extrair uma série de medidas que permitem mensurar o desempenho do modelo, sendo que a **Acurácia** é a mais utilizada para problemas de classificação.

Para cálculo da acurácia, divide-se a soma de classificações corretas (TP + TN) pelo tamanho total da amostra (TP + TN + FP + FN).

Outra forma de medição de desempenho de um modelo é o uso da função de **erro** (*loss*). Comparando-se o índice de erro entre a amostra de treino de um conjunto de dados com o índice de erro da amostra de validação, pode-se identificar o desempenho do modelo quanto à ocorrência ou não de *overfitting*. Se a taxa de erro for maior na amostra de validação, indica que está ocorrendo *overfitting*, ou seja, o modelo consegue classificar corretamente as imagens da amostra de testes, mas não consegue generalizar suficientemente para poder classificar as imagens da amostra de validação ou de teste. Nesse caso, quanto menor a taxa de erro, melhor a generalização do modelo para fins de classificação de imagens.

2.2 Trabalhos Relacionados

Em LIMA (2018) foi realizado o comparativo entre três modelos de redes neurais já existentes, aplicadas sobre um banco de dados de imagens de raças de cães (DOG, 2019), disponível no site KEAGLE (2019), tendo por objetivo identificar qual dos três modelos consegue resolver com melhores resultados um desafio proposto pelo site para identificação de imagens. Sobre o modelo original das redes foram realizados ajustes de parâmetros e aplicadas três técnicas de transformação das imagens para melhora de resultados durante a etapa de treino e comparados os resultados em cada rede neural com cada combinação das técnicas aplicadas, e também com as demais redes para identificação daquela que apresentou melhor desempenho e com qual técnica na classificação das imagens. A diferença deste trabalho com a pesquisa atual é que neste pretende-se seguir uma sequência de etapas para definir os hiperparâmetros de configuração da rede neural a fim de criar um método de análise desses hiperparâmetros que auxiliem quando se for configurar outros modelos de redes neurais convolucionais para outros formatos de imagens.

Em PEDERSETTI (2018) é proposto um modelo para auxiliar na tarefa de realizar investimentos em ações e encontrar o melhor momento de comprar ou vender ações a fim de maximizar os ganhos com estes investimentos. Para isso, foi escolhido o modelo de redes LSTM, mais adequado ao uso com séries temporais. A estrutura da rede neural utilizada foi um modelo sugerido em bibliografia quanto à quantidade de camadas. O número de neurônios de cada camada oculta foi definido mediante testes com várias combinações de mais e menos neurônios em cada camada, sendo calculado o desempenho de cada combinação e selecionado o que apresentou melhor resultado. Para a função de

ativação e a de otimização foram testadas várias fórmulas e selecionada as que apresentaram maior resultado nas predições. O que se tem em comum deste trabalho para o projeto proposto, é a utilização de um método aplicado para a definição de parâmetro da rede a fim de obter os que apresentam melhores resultados. A proposta do projeto atual é de se estender o método de testes de parâmetros a todos os parâmetros de configuração da rede, desde as funções de ativação, técnicas de regularização, taxa de aprendizado, e otimizadores, dando ao utilizador as ferramentas necessárias para construir uma rede desde o início, selecionando sempre a melhor configuração, a fim de obter o melhor resultado aplicado ao conjunto de dados objeto de estudo.

Na Tabela 1, é apresentado um comparativo entre as características principais dos trabalhos relacionados com o presente artigo.

Tabela 1 - Comparativo entre os trabalhos relacionados e o presente artigo

	LIMA	**PEDERSETTI**	**ESTE ARTIGO**
Tema	Reconhecimento Imagens	Investimentos em Ações	Reconhecimento de Imagens
Base de Dados	Dog Breed Identification	Dados de valores de ações obtidos em plataforma de investimentos criando cenários de investimento	CIFAR-10 (CIFAR10, 2019)
Modelo de Rede neural	VGG16, InceptionV3 e ResNet50, basedas em Redes Neurais Convolucionais	Modelo de Rede Neural Recorrente do tipo LSTM	Modelo básico sugerido pelo site KERAS (2019) com base em Redes Neurais Convolucionais
Método de análise	Comparativo de desempenho entre as várias configurações das tres redes	Comparativo de desempenho da rede neural com cada um dos cenários de aplicação	Otimização das configurações dos parâmetro através de uma sequência de etapas e comparativo de desempenho
Parâmetros Ajustados	Uso de técnica de aumento de dados, regularização, Otimizador, *Dropout*	Uso de técnicas de regularização	Função de Ativação, Otimizador, *Dropout*, Tamanho de Lote e Taxa de Aprendizado
Objetivo principal	Testar qual das redes propostas tem melhor desempenho	Configurar o modelo da rede ajustando os parâmetros para os cenários propostos a fim de obter os melhores resultados na indicação de investimentos.	Definir método de ajuste de parâmetros para se obter um melhor desempenho, independente do modelo de rede utilizado

Em resumo, diferente os dois trabalhos relacionados que são apresentados acima, onde um compara os modelos de redes neurais configuradas para um fim específico ou o que configura um modelo padrão de rede para uma determinada tarefa, o artigo proposto

procura analisar as configurações de uma rede neural de forma gradativa em cada um dos seus principais parâmetros de configuração, de forma a encontrar a melhor combinação destes, chegando a melhorar o resultado obtido pelo modelo original.

3 Método de Pesquisa

O presente artigo, do ponto de vista de sua natureza e segundo PRODANOV e FREITAS (2013), caracteriza-se como pesquisa aplicada, uma vez que objetiva gerar conhecimentos para a aplicação prática, dirigidos à solução de problemas de reconhecimento de imagens por computador. Dados seus objetivos, elencados acima, o trabalho de pesquisa se enquadra como uma pesquisa exploratória e experimental, onde são definidas as variáveis que influenciam no objeto de estudo, a forma de controle dessas variáveis e a medição de resultados da influência das variáveis sobre o modelo.

4 Definição do Método de Otimização

Nesta seção é apresentado o método propriamente dito, iniciando com a descrição da base de dados e das ferramentas utilizadas e, posteriormente, a descrição do método proposto, acompanhada dos resultados das análises realizadas sobre a base de dados.

4.1 Base De Dados

A base de dados escolhida para a realização deste trabalho de pesquisa foi a CIFAR10, disponível em KERAS (2019), um conjunto de 60.000 imagens coloridas com o tamanho de 32 x 32 pixels, igualmente distribuídas em 10 categorias, sendo 6.000 imagens em cada categoria.

Na Figura 7 são apresentadas algumas amostras de imagens aleatórias de cada categoria da base de dados.

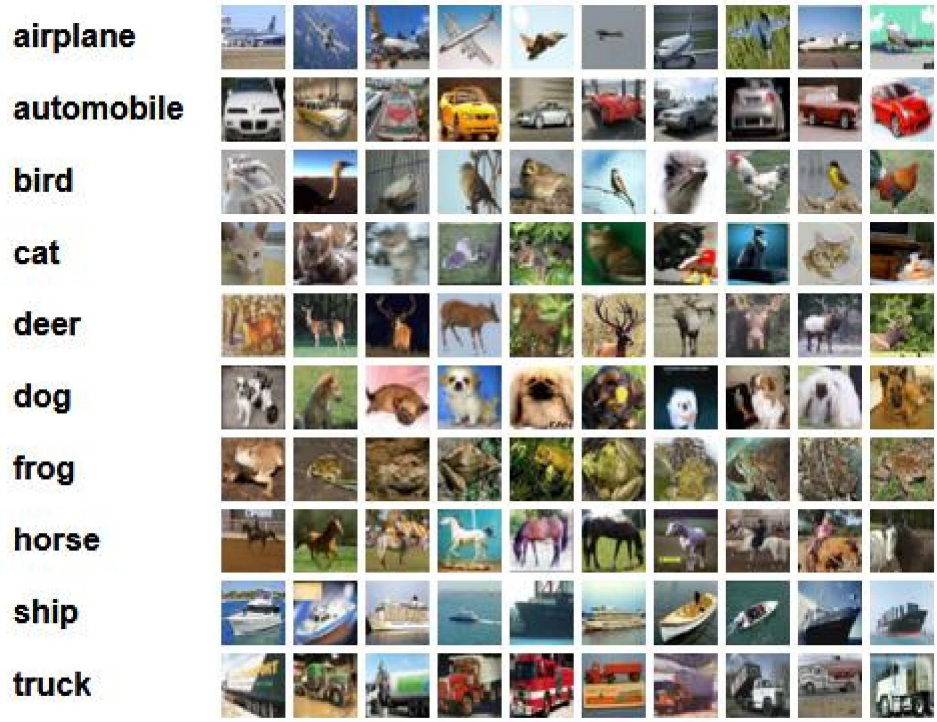

Figura 7 - Amostra de imagens aleatórias de cada categoria da base CIFAR10

4.2 Ambiente De Execução

Para a implementação das redes convolucional foi utilizada a biblioteca Keras (KERAS, 2019), que utiliza a linguagem de programação Python e usa um ambiente Júpiter Notebooks.

O ambiente Júpiter Notebooks permite a execução de códigos de forma simples e rápida em uma interface amigável integrando código e texto, de forma a permitir uma formatação das saídas visualmente clara para interpretação dos resultados.

Para rodar o código, foi escolhido o ambiente Google Colaboratory (COLAB, 2019), plataforma on-line e de acesso gratuito, que permite a execução de Notebooks Júpiter, além de disponibilizar a configuração de execução utilizando Unidades Gráficas de Processamento (Graphics Processing Unit – GPU) em um ambiente na nuvem, o que torna extremamente mais rápidos os treinos de modelos de redes neurais em comparação com a execução no próprio computador.

A biblioteca Keras é uma API de alto nível para construir e treinar modelos de *deep learning*. É usada para prototipagem rápida, pesquisa avançada e produção, com as seguintes vantagens (KERAS, 2019):

- Interface simples e consistente, otimizada para casos de uso comum, fornecendo *feedback* para erros do usuário.

- Modularidade: os modelos são feitos conectando blocos de construção configuráveis e parametrizáveis.

- Facilidade de expansão: permite a construção e agregação de novas funções e módulos para se adaptar a novas ideias de pesquisa.

4.3 Modelo Da Rede Neural Convolucional

O modelo de Rede Neural Convolucional utilizado foi baseado num modelo disponível no site da biblioteca Keras (CIFAR10CNN, 2019), onde são oferecidos alguns modelos prontos para execução.

Inicialmente foi construído um modelo exatamente como o sugerido pelo site a fim de avaliar o seu desempenho.

O modelo construído tem a seguinte estrutura:

- 2 camadas convolucionais com 32 filtros de 3x3, sem *padding*, e com função de ativação ReLU.

- 1 camada de subamostragem (*pooling*) usando filtro de 2x2 e função *max*.

- Aplicação do *dropout* de 0,25.

- Mais 2 camadas convolucionais com 64 filtros de 3x3 e demais configurações iguais às duas primeiras camadas convolucionais.

- Mais uma camada de subamostragem com filtro 2x2 e função *max*.

- Mais uma aplicação de *dropout* de 0,25.

- Uma camada *flatten* para servir de entrada para a camada densa oculta

- Uma camada densa de 512 unidades e função de ativação ReLU.

- Aplicação de *dropout* de 0,5.

- E, por fim, uma camada de saída com 10 unidades para representar as 10 categorias a serem identificadas.

O treinamento feito em 100 épocas, otimizador RMSProp com uma taxa de aprendizado de 0,0001 e com a função de erro de entropia cruzada categórica, obtendo as métricas de taxa de erro (*loss*) de 0,76 e acurácia de 75,47 na fase de testes do modelo.

A Figura 8 demonstra o gráfico com o desempenho das métricas durante a fase de treinamento do modelo original.

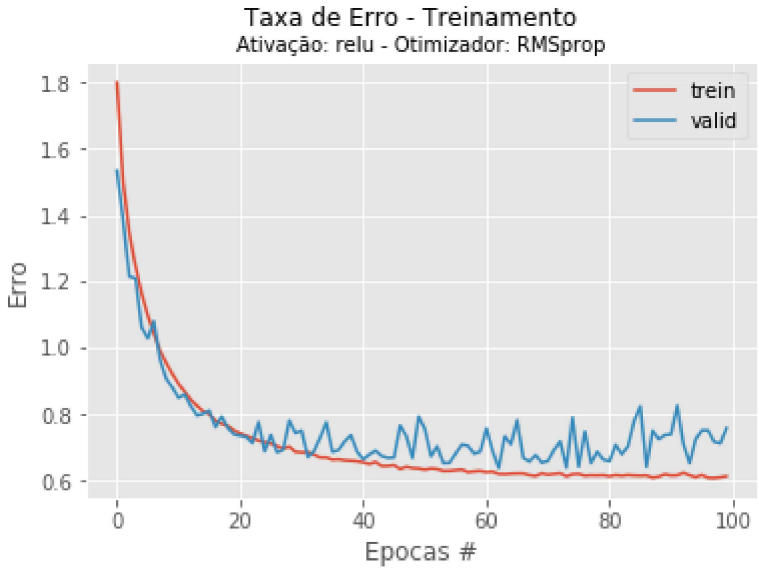

Figura 8 – Taxa de Erro dos conjuntos de Treino e Validação do modelo original

Essas métricas de desempenho, posteriormente serão comparadas com as do modelo otimizado a fim de se analisar as melhorias obtidas com a aplicação do método proposto.

4.4 Simulações

Para uma correta análise dos resultados do projeto de pesquisa, foram removidos todos os recursos de otimização e regularização da rede, como taxa de aprendizado e *dropout*, a fim de não interferirem na aplicação das técnicas de otimização dos parâmetros. Também foram reduzidas as épocas de treinamento de 100 para 25 nas primeiras etapas de simulação e, posteriormente, alteradas para 50 etapas, para agilizar a execução dos testes iniciais.

Com o modelo acima o ponto de partida da otimização do modelo tem as seguintes métricas de resultado na fase de teste:

Erro (loss): 1,42

Acurácia: 61,99

Sendo o desempenho na fase de treinamento conforme o gráfico apresentado na Figura 9.

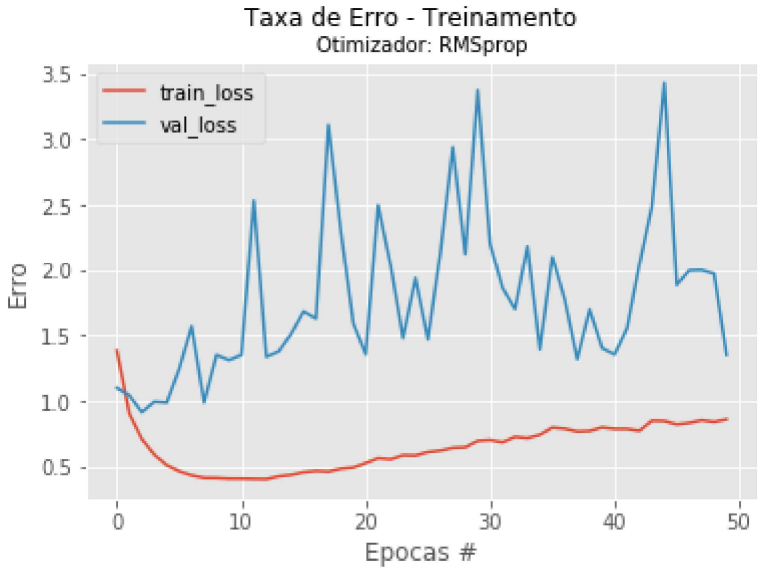

Figura 9 – Taxa de Erro dos conjuntos de Treino e Validação do modelo inicial

5 Resultados

Na Figura 10, é mostrado o esquema gráfico do método proposto, sendo que a cada etapa, são mantidas as combinações das etapas anteriores, permitindo que se visualize as diversas combinações de parâmetros para o modelo de rede neural que se está configurando, pois as escolha da melhor configuração pode variar de modelo para modelo, de acordo com as características das imagens que estiverem sendo tratadas.

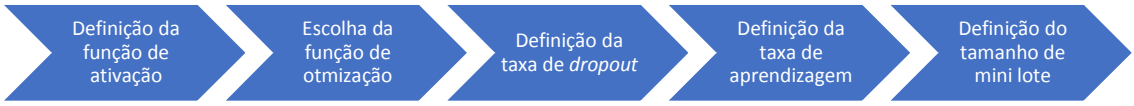

Figura 10 - Esquema Gráfico do Método

5.1 Função de Ativação e Otimizador

Foram selecionadas para teste as funções de ativação: Sigmoide, Tangente Hiperbólica e ReLU e os otimizadores SGD, RMSprop e Adam. O modelo foi alimentado com cada combinação das funções acima e foram apuradas as métricas de Erro e Acurácia do conjunto de testes para encontrar a combinação com melhor desempenho.

Nas tabelas a seguir, a coluna **Erro** e **Acurácia** se referem respectivamente à taxa de erro e à acurácia obtidos na aplicação do modelo ao conjunto de testes. As colunas **Erro Treino** e **Erro Valid** exibem a taxa de erro obtida na última época de treinamento durante a etapa de treino, com os conjuntos de treino e validação, respectivamente. Já a coluna ***Overfitting***, corresponde à diferença entre o erro do conjunto de validação e o conjunto de treino, a fim de se comparar qual configuração apresenta maior ou menor *overfitting*, ou *underfitting*, no caso de valores negativos.

Na Tabela 2 constam os resultados da execução de cada função de ativação com cada otimizador. A função Sigmoide foi a que apresentou os piores desempenhos. As funções de ativação Tanh e ReLU, por não utilizarem resultados negativos na sua saída, acabaram tendo resultados melhores. Quanto aos otimizadores, Adam foi o que teve o

melhor resultado com as funções de ativação Sigmoide e Tanh. Já, com a função de ativação ReLU, foi o otimizador RMSprop que teve melhor desempenho.

Tabela 2 - Métricas para seleção de Função de Ativação e Otimizador

Ativação	Otimizador	Erro	Acurácia	Erro Treino	Erro Valid	Overfitting
Sigmoide	SGD	2,31	10,00	2,3066	2,3066	-
Sigmoide	RMSprop	2,30	10,00	2,3027	2,3028	0,0001
Sigmoide	**Adam**	1,55	**63,37**	0,0094	1,4813	1,4719
TanH	SGD	0,85	70,51	0,5393	0,8385	0,2992
TanH	RMSprop	1,46	70,34	0,0291	1,3921	1,3630
TanH	**Adam**	1,63	**72,70**	0,0001	1,5818	1,5817
Relu	SGD	1,27	64,44	0,3678	1,2077	0,8399
Relu	**RMSprop**	2,59	**73,47**	0,0466	2,4444	2,3978
Relu	Adam	2,02	73,30	0,0328	1,8987	1,8659

5.2 Taxa de *Dropout*

Na sequência da aplicação do método de otimização, será aplicada a técnica de *dropout* com valores diferentes em cada combinação das funções de ativação e otimizadores. Foram mantidas todas as combinações para que fosse possível analisar o impacto do uso de *dropout* em cada uma delas.

As taxas de *dropout* selecionadas para análise foram as de 0.1, 0.3 e 0.5.

Na Tabela 3 pode-se observar que a função Sigmoide continuou apresentando resultados desfavoráveis, sendo que o otimizador SGD permaneceu com o mesmo desempenho sofrível de quando não tinha a aplicação de *dropout*. Já com o otimizador RMSprop, passou a apresentar melhoria de desempenho, superando os resultados obtidos com o otimizador Adam, que, por sua vez, só apresentou um resultado razoável com a menor taxa de *dropout*.

Com as demais funções de ativação, a taxa de 0.3 de *dropout* foi a que teve melhor desempenho na maioria das combinações testadas.

Tabela 3 - Métricas para seleção da taxa de *Dropout*

Ativação	Otimizador	*Dropout*	Erro	Acurácia	Erro Treino	Erro Valid	*Overfitting*
Sigmoide	SGD	0,1	2,30	10,00	2,3039	2,3029	-0,0010
Sigmoide	SGD	0,3	2,30	10,00	2,3031	2,3027	-0,0004
Sigmoide	SGD	0,5	2,30	10,00	2,3030	2,3028	-0,0002
Sigmoide	RMSprop	0,1	1,35	64,10	0,1696	1,2820	1,1124
Sigmoide	RMSprop	0,3	1,06	**65,15**	0,7264	1,0077	0,2813
Sigmoide	RMSprop	0,5	1,16	59,66	1,2337	1,1546	-0,0791
Sigmoide	Adam	0,1	1,28	**64,81**	0,1266	1,2589	1,1323
Sigmoide	Adam	0,3	2,30	10,00	2,3028	2,3027	-0,0001
Sigmoide	Adam	0,5	2,30	10,00	2,3035	2,3038	0,0003
TanH	SGD	0,1	0,85	**71,40**	0,7281	0,8222	0,0941
TanH	SGD	0,3	0,95	67,25	0,9270	0,9235	-0,0035
TanH	SGD	0,5	1,13	60,24	1,0621	1,1045	0,0424
TanH	RMSprop	0,1	1,34	71,80	0,0957	1,2722	1,1765
TanH	RMSprop	0,3	0,80	**74,56**	0,4690	0,7718	0,3028
TanH	RMSprop	0,5	0,79	72,79	0,8496	0,7614	-0,0882
TanH	Adam	0,1	1,26	71,87	0,1071	1,2148	1,1077
TanH	Adam	0,3	0,80	**74,29**	0,4845	0,7598	0,2753
TanH	Adam	0,5	0,80	71,86	0,8656	0,7730	-0,0926
Relu	SGD	0,1	0,92	**67,79**	0,7470	0,8943	0,1473
Relu	SGD	0,3	0,98	65,63	1,0342	0,9703	-0,0639
Relu	SGD	0,5	1,13	59,94	1,2437	1,1244	-0,1193

Ativação	Otimizador	*Dropout*	Erro	Acurácia	Erro Treino	Erro Valid	*Overfitting*
Relu	RMSprop	0,1	1,43	75,49	0,1537	1,2930	1,1393
Relu	**RMSprop**	**0,3**	0,84	**78,38**	0,5147	0,7700	0,2553
Relu	RMSprop	0,5	0,89	70,62	0,8242	0,8819	0,0577
Relu	Adam	0,1	1,35	75,22	0,0674	1,2988	1,2314
Relu	Adam	0,3	0,80	77,93	0,2095	0,7234	0,5139
Relu	**Adam**	**0,5**	0,62	**79,15**	0,5769	0,5903	0,0134

Das combinações acima, foram selecionadas as duas com melhor desempenho para serem levadas para a próxima etapa, com o propósito de simplificar o modelo, pois as combinações começam a ficar mais complexas, pois aumentam exponencialmente em quantidade.

Na Figura 11 é exibido o gráfico de variação da taxa de erro durante as épocas de treino para a primeira das duas combinações selecionadas. Pode-se observar que o erro no conjunto de validação se torna maior que o erro do conjunto de treino, indicando a ocorrência de *overfitting*, o que reduz a capacidade de generalização do modelo. Porém, essa ocorrência não é tão acentuada, havendo a possibilidade de se reduzir essa taxa nas próximas sequencias do método.

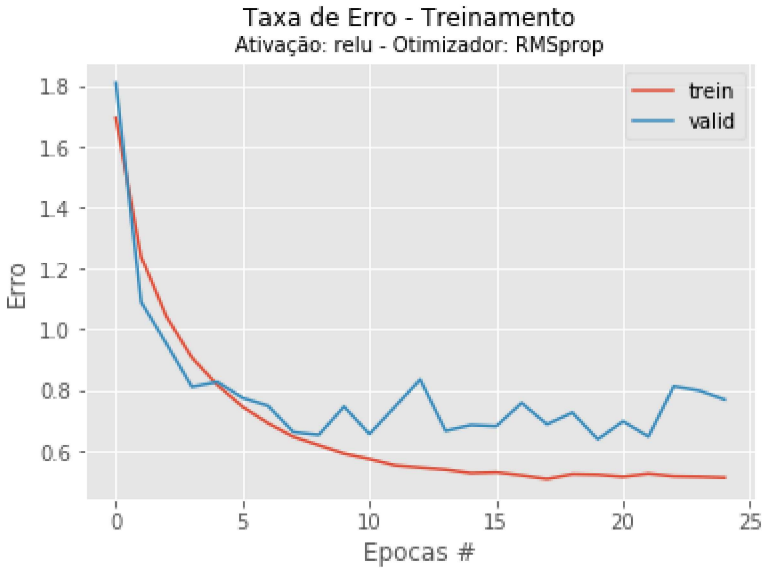

Figura 11 - Taxa de Erro durante o treino da primeira combinação selecionada

Na Figura 12, é exibido o gráfico de erro para a segunda combinação selecionada, demonstrando um *overfitting* bem inferior, garantindo uma melhor generalização.

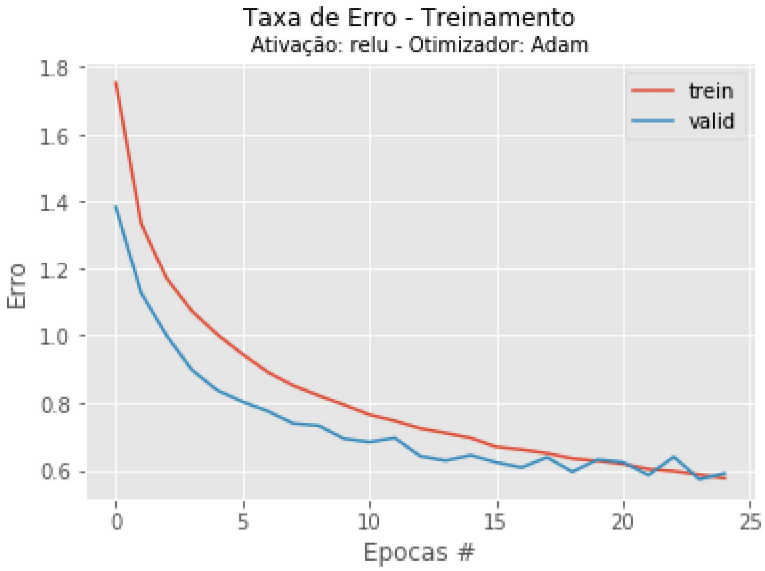

Figura 12 - Taxa de Erro durante o treino da segunda combinação selecionada

5.3 Taxa de Aprendizado e Mini Lote

Para a análise da taxa de aprendizado e tamanho de mini lote, foram selecionados apenas as duas combinações de melhor desempenho da etapa anterior, como já explicado acima.

Nessa etapa, foram selecionadas as taxas de aprendizado de 0.0001, 0.0005 e 0.001 e os tamanhos de mini lote de 32, 64, 128 e 256. As taxas de aprendizado foram combinadas com cada tamanho de mini lote e, com cada uma das combinações escolhidas da etapa anterior e apresentadas na Tabela 4.

Nas simulações realizadas, as combinações utilizando o otimizador Adam mantiveram os melhores resultados, melhorando ainda mais o desempenho com a aplicação de variações das taxas de aprendizado e de tamanhos de mini lotes.

Tabela 4 - Métricas para seleção da Taxa de Aprendizado e tamanho do mini lote

Ativação	Otimizador	Dropout	Tx Aprendiz.	Tam Lote	Erro	Acurácia	Erro Treino	Erro Valid	Overfitting
Relu	RMSprop	0,3	0,0001	32	0,68	78,75	0,5497	0,6569	0,1072
Relu	RMSprop	0,3	0,0001	64	0,64	79,18	0,4117	0,6052	0,1935
Relu	RMSprop	0,3	0,0001	128	0,68	77,48	0,4003	0,6543	0,2540
Relu	RMSprop	0,3	0,0001	256	0,73	74,72	0,5878	0,7065	0,1187
Relu	RMSprop	0,3	0,0005	32	1,00	67,51	1,1152	0,9973	-0,1179
Relu	RMSprop	0,3	0,0005	64	0,88	74,54	0,5378	0,8587	0,3209
Relu	RMSprop	0,3	0,0005	128	0,98	79,14	0,2964	0,9065	0,6101
Relu	RMSprop	0,3	0,0005	256	1,01	77,51	0,1446	0,9254	0,7808
Relu	RMSprop	0,3	0,001	32	14,51	10,00	14,4912	14,6352	0,1440
Relu	RMSprop	0,3	0,001	64	1,48	70,35	0,9845	1,4304	0,4459
Relu	RMSprop	0,3	0,001	128	0,78	78,69	0,4523	0,7289	0,2766
Relu	RMSprop	0,3	0,001	256	0,90	77,41	0,1900	0,8745	0,6845
Relu	Adam	0,5	0,0001	32	0,59	79,76	0,4421	0,5599	0,1178
Relu	Adam	0,5	0,0001	64	0,64	77,69	0,5454	0,6187	0,0733
Relu	Adam	0,5	0,0001	128	0,70	75,97	0,6591	0,6704	0,0113
Relu	Adam	0,5	0,0001	256	0,75	73,76	0,7849	0,7290	-0,0559

Ativação	Otimizador	Dropout	Tx Aprendiz.	Tam Lote	Erro	Acurácia	Erro Treino	Erro Valid	Overfitting
Relu	Adam	0,5	0,0005	32	0,56	81,92	0,3769	0,5341	0,1572
Relu	Adam	0,5	0,0005	64	0,59	80,50	0,3588	0,5635	0,2047
Relu	Adam	0,5	0,0005	128	0,58	80,81	0,3516	0,5271	0,1755
Relu	Adam	0,5	0,0005	256	0,58	81,16	0,3849	0,5518	0,1669
Relu	Adam	0,5	0,001	32	0,65	78,23	0,6003	0,6292	0,0289
Relu	Adam	0,5	0,001	64	0,60	79,76	0,4762	0,5562	0,0800
Relu	**Adam**	**0,5**	**0,001**	**128**	0,58	81,03	0,4334	0,5449	0,1115
Relu	Adam	0,5	0,001	256	0,58	80,62	0,3967	0,5518	0,1551

Das combinações obtidas, a que foi selecionada como a de melhor desempenho (em destaque na Tabela 4) apesar de não ter o maior valor da métrica de Acurácia, tem a menor taxa de erro e uma baixa ocorrência de *overfitting*.

Na Figura 13 é apresentado, à esquerda, o gráfico do modelo original que serviu de base para a aplicação do método de otimização e, à direita, o modelo otimizado.

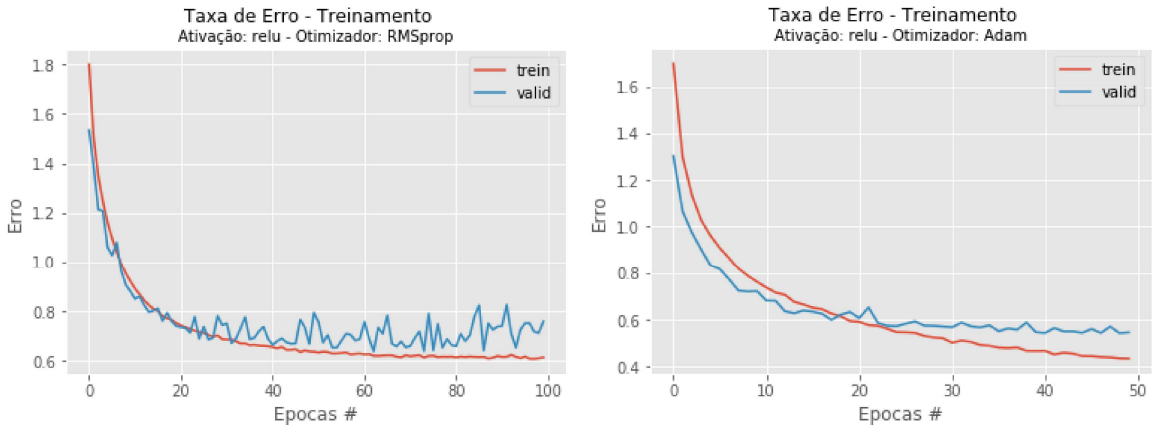

Figura 13 - Taxas de erro do modelo original e do modelo otimizado

5.4 Discussão dos Resultados

As melhorias de desempenho do modelo são apresentadas na Tabela 5, onde se pode observar que, apesar do pouco incremento na acurácia, houve uma redução significativa das taxas de erro e de *overfitting*, garantindo uma maior generalização do modelo.

Tabela 5 - Comparativo de Métricas entre o modelo original e o modelo otimizado

		Modelo Original	Modelo Otimizado Pelo Método	Melhorias
Etapa de Teste	Acurácia	75,47	81,03	7,37%
	Erro	0,76	0,58	-23,68%
Etapa de Treinamento	Erro - Treinamento	0,6124	0,4334	-29,23%
	Erro - Validação	0,7571	0,5449	-28,03%
	Overfitting	0,1447	0,1115	-22,94%

A evolução de resultados obtidos pelo método procura manter as faixas de valores de etapas anteriores, a fim de permitir a avaliação dos mesmos parâmetros em conjunto com as novas otimizações adicionadas a cada etapa. Como exemplo, na primeira etapa onde foram testadas as funções de ativação, de acordo com a Tabela 2, a função de ativação RMSprop foi a que apresentou uma acurácia mais elevada, apesar de ter as mais altas taxas de erro também. Mas à partir da etapa seguinte, onde se aplica a técnica de

dropout, e com um índice específico, a função de ativação Adam se destaca com a maior acurácia e uma baixa taxa de erro, como pode ser visto na Tabela 3.

Por fim, conforme descrito por SMITH et al (2018), antes de se reduzir a taxa de aprendizado, deve-se procurar aumentar o tamanho do mini lote, sempre analisando o contexto onde está sendo aplicado o teste. Nesse caso, pode-se observar que os melhores resultados em cada configuração, apresentaram-se entre as combinações com maiores tamanhos de mini lote, como pode ser observado nos resultados apresentados na Tabela 4.

6 Conclusão

O presente artigo propôs um método de análise de hiperparâmetros de configuração de uma rede neural convolucional para reconhecimento de imagens, de forma a permitir a escolha das combinações com melhores resultados para a construção do modelo. De acordo com os valores das métricas obtidas no final da análise, pode-se concluir que o método é válido, tanto que permitiu que se chegasse a uma melhoria no desempenho final do modelo com relação ao modelo original, aumentando em mais de 7% a acurácia do modelo e reduzindo em mais de 20% as taxas de erro e de ocorrência de *overfitting*, aumentando o grau de generalização do modelo. Com isso, garante-se um melhor resultado do uso do modelo para classificação de novas imagens do mesmo padrão que não tenham sido submetidas ainda ao modelo durante as etapas de treinamento e testes.

Neste trabalho foi utilizada uma estrutura de rede neural pronta, e foram realizadas simulações com os principais parâmetros de configuração até se chegar em um modelo de configuração ideal. Trabalhos futuros podem incluir a aplicação do método a outras estruturas de redes neurais para otimizá-las com relação às suas configurações originais. Também pode ser estendido o método para incluir o estudo da quantidade de camadas e da quantidade de unidades de processamento por camada, permitindo tornar uma rede neural mais densa, chegando a um maior nível de generalização, sempre integrado com os demais parâmetros estudados.

Referências

BEHNKE, Sven. **Hierarchical Neural Networks for Image Interpretation**. Lecture Notes in Computer Science. Vol. 2766. Berkeley. 2003.

BISHOP, Christopher M. **Pattern Recognition and Machine Learnig**. Springer Science+Business Media. Singapore. 2006.

CIFAR10CNN. Disponível em: <https://keras.io/examples/cifar10_cnn/>. Acesso em: julho, 2019.

COLAB. Disponível em: <https://colab.research.google.com/notebooks/welcome.ipynb>. Acesso em julho, 2019.

Deep Learning Book, 2019. Data Science Academy. Disponível em: http://www.deeplearningbook.com.br. Acesso em: julho. 2019.

DOG Breed Identification. Disponível em: <https://www.kaggle.com/c/dog-breed-identification>. Acesso em: junho, 2019.

FLACH, Peter. **Machine Learning. The Art and Science of Algorithms that Make Sense of Data**. Cambridge, 2012

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. MIT Press. http://www.deeplearningbook.org. 2016;

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Hamilton, Canada: Pearson, 2008.

KERAS. Disponível em: <https://keras.io/>. Acesso em: julho, 2019.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 1998.

LIMA, Francisco Igor Da Silva. **Estudo Comparativo Entre Redes Neurais Convolucionais Para Um Problema De Classificação**. UFCE. Quixadá. 2018.

LUGER, G. F. **Inteligência Artificial**. 6. ed. São Paulo, Brasil: Pearson, 2013.

MCCULLOCH, W. S.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics 5, 1943

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. **Activation Functions: Comparison of trends in Practice and Research for Deep Learning**. CoRR, v. abs/1811.0, 2018.

PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioner's Approach**. O'Reilly Media. CA. 2017.

PEDERSETTI, Eduardo J. **Predições de Preços em Mercado de Ações - Um Estudo de Caso Aplicando Redes Neurais Long Short-Term Memory e Indicadores Técnicos em um Ambiente Simulado**. UNISINOS. São Leopoldo. 2018.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico**. 2. ed. Novo Hamburgo. Feevale. 2013.

RUDER, S. **An overview of gradient descent optimization algorithms**. CoRR, v. abs/1609.0, 2016.

SNOEK J.; LAROCHELLE, H.; ADAMS, R. **Practical Bayesian Optimization of Machine Learning Algorithms**. Advances in Neural Information Processing Systems. Curran Associates, Inc., 2012.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. Journal of Machine Learning Research, 2014.

TENSORFLOW. Disponível em: <https://www.tensorflow.org/resources/models-datasets>. Acesso em: julho, 2019.

YANN, L.; BENGIO, Y.; HINTON, G. **Deep Learning**. Nature. 2015.