28/6/2018

Trabalho Final PTC3569

Relatório - Estudo de redes neurais aplicadas a Classificação de 3 classes de animais utilizando Deep Learning

Pedro Shinzato
8660151

Sumário

1 Introdução

1.1 Motivações

Deep Learning tem sido um assunto crescente tanto no meio da pesquisa acadêmica, quanto no meio de empresas. Parte disso se deve ao progresso realizado na área da eletrônica que possibilitou processadores mais poderosos e memórias cada vez maiores, capazes de processar grandes quantidades de dados. Gigantes como Google e Facebook têm utilizado cada vez mais aplicações que utilizam machine learning, uma vez que um ponto fundamental do modelo de negócios dessas empresas é a utilização dos dados obtidos de seus usuários. O grande volume de dados possibilita o oferecimento de serviços de AI (*Artificial Inteligence*) cada vez mais precisos, como o Google duplex apresentado este ano, serviço que utiliza uma máquina para ligar para um estabelecimento e marcar uma reserva (Google Duplex Demo from Google IO 2018).

A popularização de aplicações desse tipo desperta interesse de muitas pessoas e levam muitos a confiar no sólido futuro que essa área terá nos próximos anos. O número de pesquisas publicadas com a palavra-chave Deep Learning dobrou de 2016 para 2017, foi de 5.131 para 10.647, (Scopus, 2018).

Nesse contexto, decidiu-se realizar esse trabalho nessa área, mais especificamente voltado para visão computacional.

1.2 Objetivo

O objetivo desse trabalho é desenvolver conceitos de deep learning, convolutional neural networks (CNNs), visão computacional (*computer vision*) aplicados a um problema de classificação utilizando uma rede neural de estrutura simples. Os resultados da solução serão comparados com redes mais complexas para uma análise qualitativa do problema.

2 Revisão de Conceitos

2.1 Deep Learning

“Deep learning 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 descoberta de drogas e genômica. O aprendizado profundo descobre estruturas complexas em grandes conjuntos de dados usando o algoritmo de Backporpagation para indicar como uma máquina deve alterar seus parâmetros internos que são usados para calcular a representação em cada camada da representação na camada anterior. Redes convolucionais profundas trouxeram avanços no processamento de imagens, vídeo, fala e áudio, enquanto redes recorrentes iluminaram dados sequenciais como texto e fala”, (Lecun, Hinton, & Bengio, 2015).

Segundo o site da MathWorks, desenvolvedora do software MATLAB, Deep Learning é a técnica de Machine Learning que ensina a computadores a fazerem o que é natural aos humanos: aprender por exemplo. Deep Learning é a tecnologia chave que possibilita que carros autônomos entendam placas, saibam diferenciar um poste de um pedestre. Essa

tecnologia está atingindo resultados nunca antes vistos, com precisão algumas vezes superior à de humanos. Esse modelo se utiliza de redes neurais com muitas camadas e é treinado por uma grande quantidade de dados.

2.2 Redes Convolucionais

São redes que executam a operação de convolução discreta em duas dimensões, muito utilizadas para extrair as principais características de uma imagem por meio de filtros.

A imagem é convolucionada com um filtro gerando um mapa de atributos, depois esse mapa de atributos é passado por uma camada não linear, geralmente uma camada reLU que faz com que os valores negativos se tornem zero. Após isso, ela passa por uma operação de pooling que diminiu a dimensão dos dados trabalhados (downsampling), reduzindo custos computacionais e deixa apenas os dados mais relevantes otimizando o treinamento da rede.

No final a rede pode passar por uma camada de dropout (Dropout: A Simple Way to Prevent Neural Networks from, 2014) para reduzir as chances de overfitting no treinamento da rede seguida de uma camada fullyconnected que é responsável por determinar qual o valor de saída para cada classe. Esse resultado é normalizado por uma camada Softmax que dá como resultado as probabilidades, calculadas pela rede e correspondentes a cada classe, de uma imagem estar nessa classe.

As principais ideias por trás de redes convolucionais são:

1. Conexões locais (esparsas);
2. Pesos compartilhados (Shared Weights)
3. Operações de Pooling
4. Uso de muitas camadas

(Lecun, Hinton, & Bengio, 2015)

Essas ideias permitem um menor custo computacional (itens 1, 2 e 3) e uma análise mais detalhada dos atributos da imagem (item 4).

3 Problema proposto

Este trabalho propôs buscar uma solução para a classificação de animais visando familiarização com os conceitos acima por meio de uma atividade prática.

Antes de partir para a classificação de animais e buscando entender melhor os conceitos de redes convolucionais (CNNs), encontrei um exemplo disponibilizado pela MathWorks: Create Simple Deep Learning Network for Classification, (MathWorks, 2018).

3.1 Exemplo MATLAB – Identificação de números

Este exemplo mostra como criar e treinar uma CNN simples para reconhecimento de imagens em ambiente MATLAB. Os principais conhecimentos adquiridos foram:

- Pré-processar imagens para classificação;

- Carregar e separar imagens de treino e validação por classe;
- Definir a arquitetura de uma rede;
- Especificar opções de treino;
- Treinar a rede;
- Classificar novos dados e calcular a precisão.

O código do exemplo está no apêndice do relatório.

3.1.1 Carregamento dos dados

O exemplo se utiliza de um banco e dados com 10000 imagens de dígitos de 0 a 9. Cada classe possui 1000 imagens de 28x28 pixels em preto e branco.

Primeiramente as imagens são carregadas e categorizadas de acordo com a pasta em que estão contidas.

Depois, o programa mostra 20 imagens escolhidas aleatoriamente para ilustrar os dados que serão trabalhados.

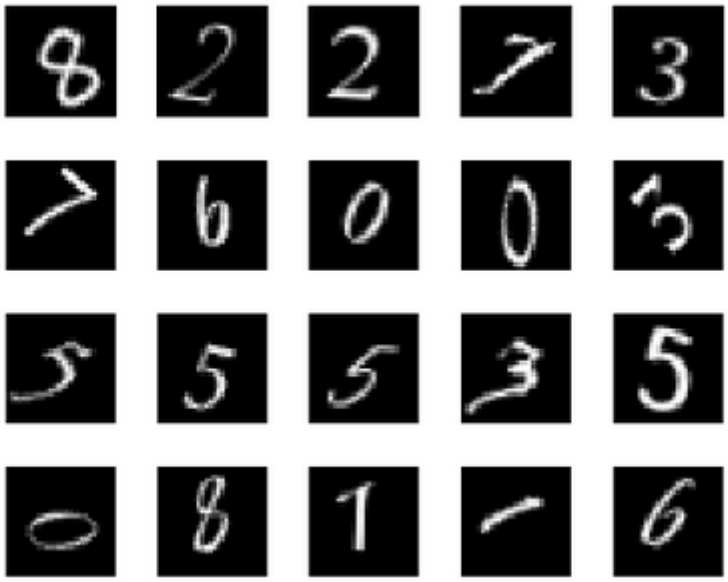


Figura 1- Display aleatório de números do dataset

O número de imagens escolhidas para treinamento foi de 750 por classe, restando 250 que são utilizadas para validação.

3.1.2 Arquitetura da Rede

ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	imageinput 28x28x1 images with 'zerocenter' normalization	Image Input	28×28×1	-
2	conv_1 8 3x3x1 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×8	Weights 3×3×1×8 Bias 1×1×8
3	batchnorm_1 Batch normalization with 8 channels	Batch Normalization	28×28×8	Offset 1×1×8 Scale 1×1×8
4	relu_1 ReLU	ReLU	28×28×8	-
5	maxpool_1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	14×14×8	-
6	conv_2 16 3x3x8 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×16	Weights 3×3×8×16 Bias 1×1×16
7	batchnorm_2 Batch normalization with 16 channels	Batch Normalization	14×14×16	Offset 1×1×16 Scale 1×1×16
8	relu_2 ReLU	ReLU	14×14×16	-
9	maxpool_2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	7×7×16	-
10	conv_3 32 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	7×7×32	Weights 3×3×16×32 Bias 1×1×32
11	batchnorm_3 Batch normalization with 32 channels	Batch Normalization	7×7×32	Offset 1×1×32 Scale 1×1×32
12	relu_3 ReLU	ReLU	7×7×32	-
13	fc 10 fully connected layer	Fully Connected	1×1×10	Weights 10×1568 Bias 10×1
14	softmax softmax	Softmax	1×1×10	-
15	classoutput crossentropyex	Classification Output	-	-

Figura 2- Arquitetura da rede de classificação de dígitos

Abaixo está a descrição das camadas e como devem ser implementadas em MATLAB.

Camada de entrada: Uma camada de entrada é onde define-se a especificação do tamanho da imagem, que, nesse caso, é de 28 por 28 por 1. Esses números correspondem à altura, largura e tamanho do canal. Os dados de dígitos consistem em imagens em escala de cinza, de modo que o tamanho do canal (canal de cores) é 1. Para uma imagem colorida, o tamanho do canal é 3, correspondendo aos valores RGB. Não é preciso embaralhar os dados porque trainNetwork, por padrão, embaralha os dados no início do treinamento.

Camada convolucional: Na camada convolucional, o primeiro argumento é o tamanho do filtro, filterSize, que é a altura e a largura dos filtros que a função de treinamento usa durante o processamentodas imagens. Neste exemplo, o número 3 indica que o tamanho do filtro é 3 por 3. Pode-se especificar tamanhos diferentes para a altura e a largura do filtro. O segundo argumento é o número de filtros, numFilters, que é o número de neurônios que se conectam à mesma região da entrada. Este parâmetro determina o número de mapas de atributos. O valor do 'Padding' é usado para adicionar preenchimento ao mapa de recursos de entrada. Para um tamanho de filtro de 3, um preenchimento de 1 garante que o tamanho da saída espacial seja o mesmo que o tamanho da entrada. Também pode-se definir o passo e as taxas de aprendizado para essa camada usando argumentos nome-valor de convolution2dLayer.

Camada de Normalização em batelada: Camadas de normalização em batelada normalizam as ativações e gradientes que se propagam através de uma rede, tornando o treinamento de rede um problema de otimização mais fácil. Deve-se utilizar camadas de normalização em batelada entre camadas convolucionais e não-linearidades, como camadas ReLU, para acelerar o treinamento da rede e reduzir a sensibilidade à inicialização da rede. O batchNormalizationLayer serve para criar uma camada de normalização em batelada.

Camada ReLU: A camada de normalização do lote é seguida por uma função de ativação não linear. A função de ativação mais comum é a unidade linear retificada (ReLU). Deve-se usar reluLayer para criar uma camada ReLU.

Camada Max Pooling: Camadas convolucionais (com funções de ativação) às vezes são seguidas por uma operação de amostragem reduzida que reduz o tamanho espacial do mapa de recursos e remove informações espaciais redundantes. Down-sampling torna possível aumentar o número de filtros em camadas convolucionais mais profundas sem aumentar a quantidade necessária de computação por camada. Uma maneira de reduzir a amostragem é usar uma operação de pool máximo, que se cria usando maxPooling2dLayer. A camada de agrupamento máximo retorna os valores máximos de regiões retangulares de entradas, especificados pelo primeiro argumento, poolSize. Neste exemplo, o tamanho da região retangular é [2,2]. O argumento de par de nome e valor 'Stride' especifica o tamanho do passo que a função de treinamento executa ao varrer a entrada.

Camada Totalmente Conectada: As camadas de convolução e de down-sampling são seguidas por uma ou mais camadas totalmente conectadas. Como o próprio nome sugere, uma camada totalmente conectada é uma camada na qual os neurônios se conectam a todos os neurônios da camada anterior. Essa camada combina todos os recursos aprendidos pelas camadas anteriores na imagem para identificar

3.1.3 Especificação dos parâmetros de treinamento

Os parâmetros de treinamento foram:

- Número máximo de Épocas: 4
- Dados de validação: 250
- Frequência de validação: 30 iterações
- Taxa de aprendizado: 0.01

3.1.4 Resultados do treinamento

A precisão obtida pela rede foi de 99%. Apesar desse valor ser bem alto, o tempo de treinamento foi de apenas 36 segundos, realizando 4 épocas, 232 iterações no total e utilizando uma única CPU. O processador é Intel Core i7-3630QM 2.40Ghz.

Como podemos observar, esse desempenho se deve a alguns fatores como a baixa resolução das imagens (28x28), estarem em escala cinza reduzindo de 3 canais, RGB, para 1 canal e a simplicidade do dataset que não possui atributos muito complexos com figuras de animais e de pessoas.

3.2 Identificação de animais

O problema proposto para este trabalho é o de classificação de 3 tipos de animais: gato, cachorro e macaco. A base de dados é composta por 1000 imagens de cada classe de tamanhos diferentes.

Por meio desse problema, são propostas soluções com diferentes arquiteturas de redes neurais que serão testadas. Objetivo desses testes é realizar uma comparação entre cada tipo de arquitetura para discutir a influência das técnicas utilizadas nelas. As soluções propostas são:

1. Utilizar a mesma arquitetura da rede de classificação de dígitos;
2. Utilizar uma arquitetura similar à rede de classificação de dígitos;
3. Utilizar Transfer Learning a partir da GoogLeNet;

São analisadas a seguir a performance de cada rede.

3.2.1 Resultados da rede do exemplo

Os resultados obtidos usando o mesmo código que o utilizado no exemplo não obteve bons resultados. Foram realizadas 10 épocas com 170 iterações no total e a precisão obtida foi de 49%. O que nos indica que a rede foi capaz de aprender um pouco sobre as classes apresentadas, já que o valor acima da precisão obtida por uma escolha aleatória (33,3%), considerando que o número de classes é 3.

Esse resultado é compreensível dado que a rede havia sido projetada para entradas diferentes: imagens pequenas 28x28. As imagens utilizadas eram maiores, 100x100 e também em preto e branco. As imagens foram pré-processadas para essa configuração para estarem mais próximas das características das imagens do exemplo.

Também foi realizado um treino com imagens 224x224 coloridas. O resultado foi igual ao caso de uma classificação aleatória 33,3%. Também é compreensível, uma vez que as imagens são muito maiores e nenhuma modificação foi feita na arquitetura da rede.

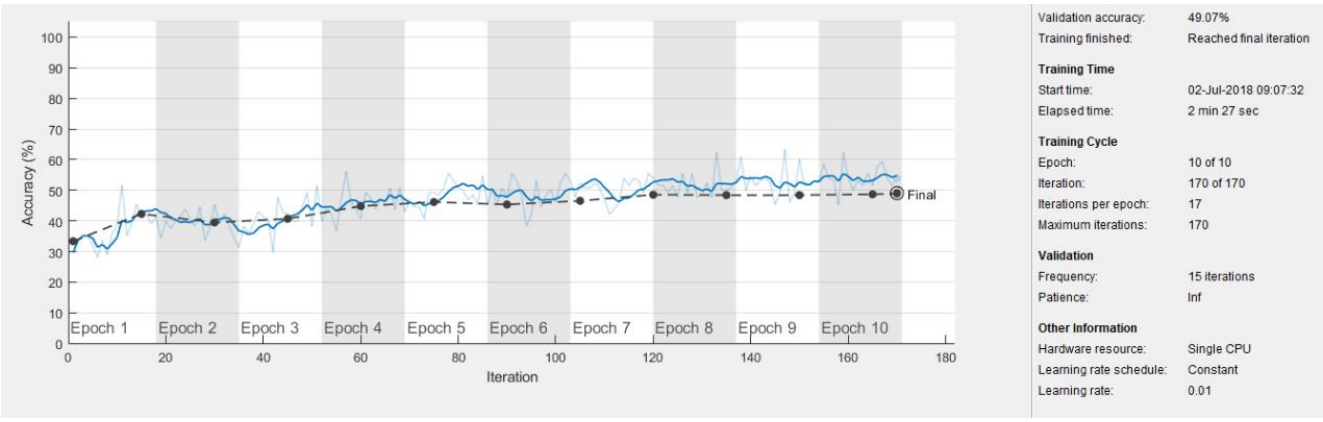
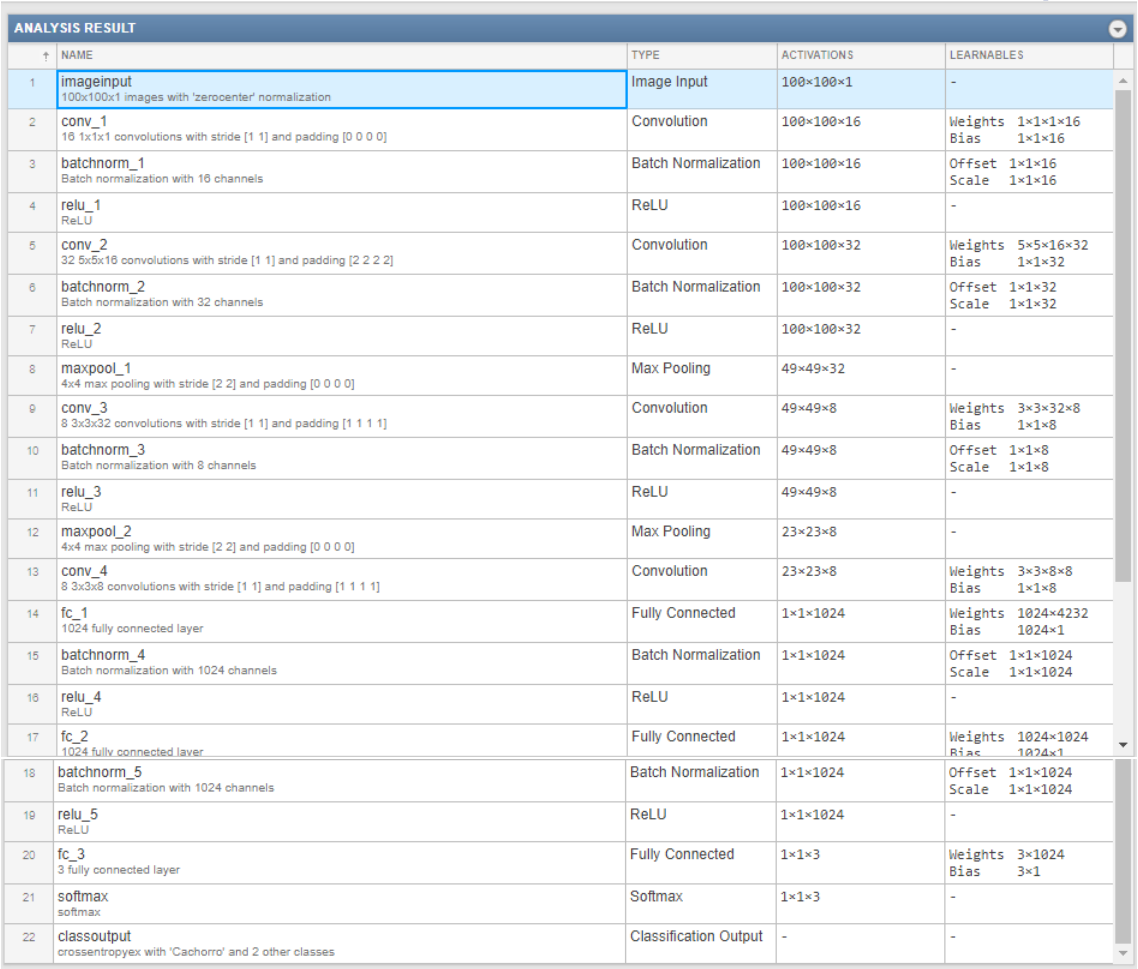


Figura 3 – Treinamento usando a rede do exemplo

3.2.2 Resultados da rede com arquitetura similar ao do exemplo

A arquitetura do item anterior é relativamente simples, porém não obteve resultados de precisão satisfatórios. Tendo em vista um melhoramento nos resultados decidiu-se implementar a arquitetura da figura abaixo, uma vez que as imagens trabalhadas são mais complexas e maiores que as do exemplo do MATLAB.

A arquitetura, porém, ainda não é tão complexa quanto a estrutura que será apresentada no item seguinte, uma vez que é uma rede de convoluções sequenciais. As principais mudanças foram a adição de mais uma camada de convolução e mais duas camadas fullyconected.



ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	imageinput 100x100x1 images with 'zerocenter' normalization	Image Input	100×100×1	-
2	conv_1 16 1x1x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	100×100×16	Weights 1×1×1×16 Bias 1×1×16
3	batchnorm_1 Batch normalization with 16 channels	Batch Normalization	100×100×16	Offset 1×1×16 Scale 1×1×16
4	relu_1 ReLU	ReLU	100×100×16	-
5	conv_2 32 5x5x16 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	100×100×32	Weights 5×5×16×32 Bias 1×1×32
6	batchnorm_2 Batch normalization with 32 channels	Batch Normalization	100×100×32	Offset 1×1×32 Scale 1×1×32
7	relu_2 ReLU	ReLU	100×100×32	-
8	maxpool_1 4x4 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	49×49×32	-
9	conv_3 8 3x3x32 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	49×49×8	Weights 3×3×32×8 Bias 1×1×8
10	batchnorm_3 Batch normalization with 8 channels	Batch Normalization	49×49×8	Offset 1×1×8 Scale 1×1×8
11	relu_3 ReLU	ReLU	49×49×8	-
12	maxpool_2 4x4 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	23×23×8	-
13	conv_4 8 3x3x8 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	23×23×8	Weights 3×3×8×8 Bias 1×1×8
14	fc_1 1024 fully connected layer	Fully Connected	1×1×1024	Weights 1024×4232 Bias 1024×1
15	batchnorm_4 Batch normalization with 1024 channels	Batch Normalization	1×1×1024	Offset 1×1×1024 Scale 1×1×1024
16	relu_4 ReLU	ReLU	1×1×1024	-
17	fc_2 1024 fully connected layer	Fully Connected	1×1×1024	Weights 1024×1024 Bias 1024×1
18	batchnorm_5 Batch normalization with 1024 channels	Batch Normalization	1×1×1024	Offset 1×1×1024 Scale 1×1×1024
19	relu_5 ReLU	ReLU	1×1×1024	-
20	fc_3 3 fully connected layer	Fully Connected	1×1×3	Weights 3×1024 Bias 3×1
21	softmax softmax	Softmax	1×1×3	-
22	classoutput crossentropyex with 'Cachorro' and 2 other classes	Classification Output	-	-

Figura 4 - Arquitetura simples

O resultado obtido foi significativamente melhorado durante o treinamento da nova rede. A precisão foi aumentada em 20% indo para 70%, e pode-se considerar satisfatório devido ao baixo tempo de processamento e a simplicidade da arquitetura da rede.

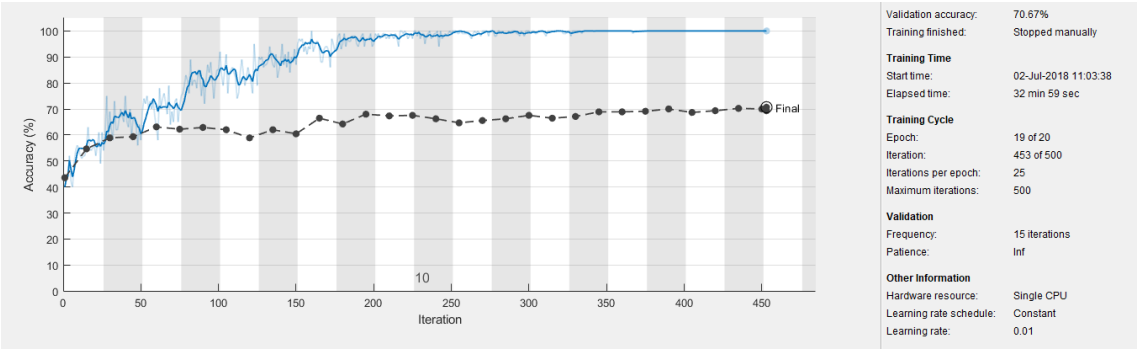


Figura 5- Treinamento da rede de arquitetura simples

Como pode-se ver na imagem, ocorre overfitting ao longo do treinamento

3.2.3 Resultados da rede GoogLenet retreinada para o problema

Utilizamos a rede pré-treinada GoogLeNet, que utiliza uma arquitetura complexa. Essa rede se possui inception modules (Going deeper with convolutions, 2015), que são estruturas de rede que realizam convoluções em paralelo com filtros de diferentes tamanhos e otimizam o

treinamento reduzindo o custo computacional por meio de filtros 1x1 que reduzem a profundidade dos dados processados.

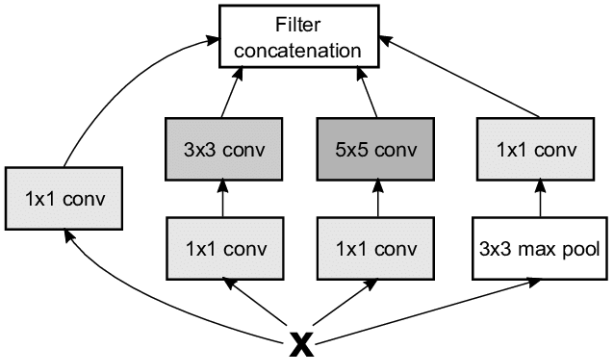


Figura 6 - Módulo de Inception

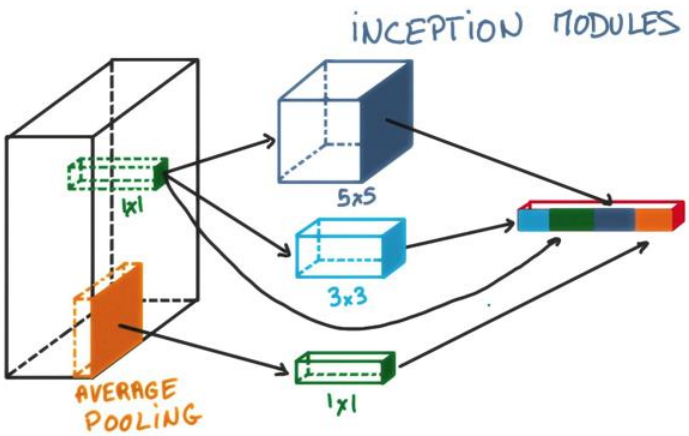


Figura 7 - Concatenação dos mapas de atributos

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figura 8 - Arquitetura GoogLeNet

A rede GoogLeNet possui 144 camadas. Para o treinamento, porém, é necessário apenas retreinar as últimas camadas, pois elas são responsáveis por atributos de maior complexidade das imagens. A figura abaixo mostra as camadas retreinadas.

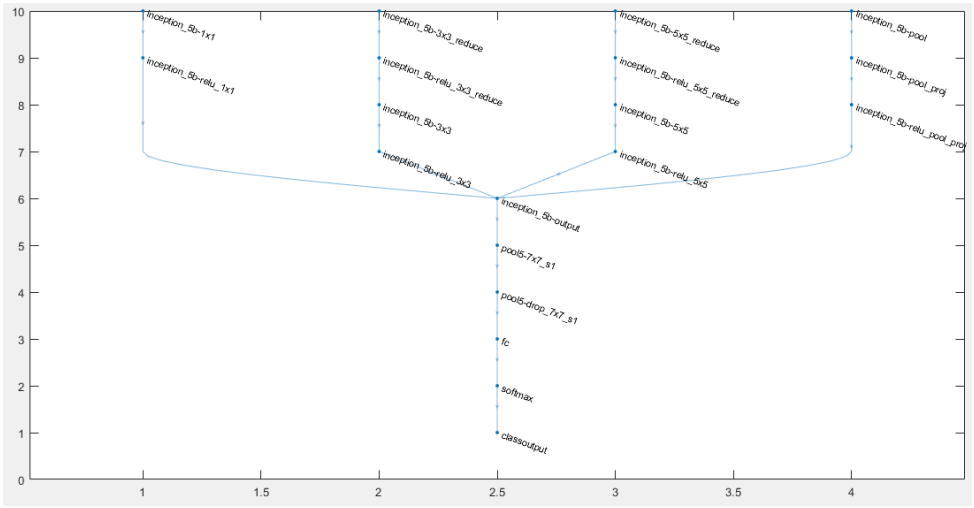


Figura 9- Camadas que foram retreinadas

Cada nó corresponde a uma camada que é retreinada. O código está anexado ao relatório e foi baseado no código da MathWorks (Transfer Learning Using GoogLeNet, 2018).

Como podemos ver, a rede é treinada com total êxito, 100% de precisão na validação. Além disso é treinada mais rapidamente que as redes mais simples e com uma base de dados menor: 150 imagens por classe, sendo 113 de treino e 37 de validação.

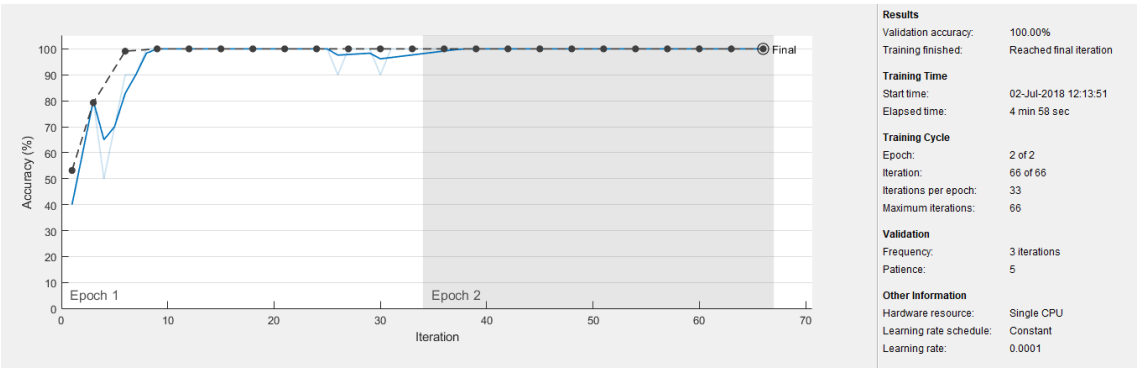


Figura 10 - Treinamento por Transfer Learning

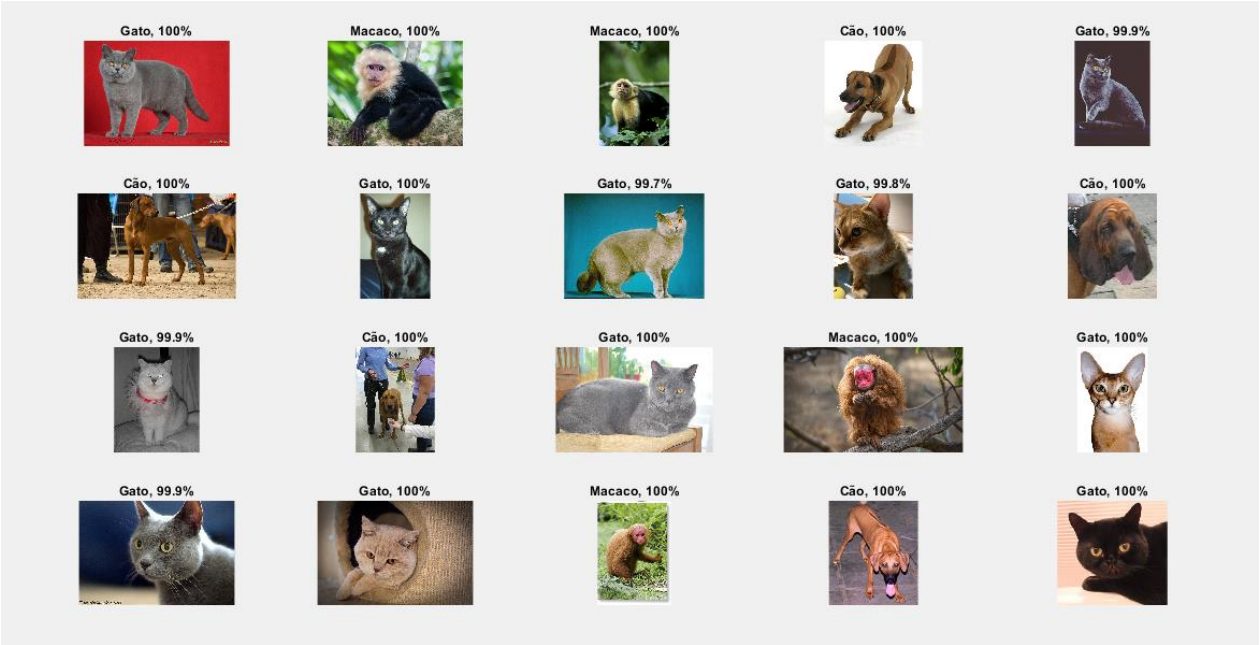


Figura 11 - Teste de imagens classificadas

4 Desafios

Os principais desafios foram:

- O tempo escasso para a realização do trabalho;
- A falta de conhecimento prévio sobre o assunto;
- A falta de poder computacional para realizar treinamento de redes mais complexas;
- O grande tempo gasto acreditando na possibilidade de melhor muito a precisão com uma rede pequena e simples se comparada a redes de grande precisão;

5 Conclusões

Redes neurais para de arquitetura simples podem ser treinadas em curto período de tempo, porém é um trade-off de entre tempo e precisão uma vez que seria necessário grande poder computacional e grande volume de dados para treinar rede mais complexas e com altíssima precisão. Como é o caso da GoogLeNet que foi treinada durantes dias, e mesmo assim considerado um treinamento rápido pelo seu tamanho.

Apesar disso a precisão foi satisfatório em virtude do contexto, sendo que o objetivo era uma maior familiaridade com o assunto.

Se tratando de um problema que busca precisão e resultados eficientes a melhor alternativa é a utilização de Transfer Learning.

6 Referências

(28 de 06 de 2018). Fonte: Scopus: https://www.scopus.com

Google Duplex Demo from Google IO 2018. (28 de 06 de 2018). Fonte: Youtube: https://www.youtube.com/watch?v=bd1mEm2Fy08

Lecun, Y., Hinton, G., & Bengio, Y. (28 de maio de 2015). Deep learning. *Nature 521*, pp. 436-444.

MathWorks. (28 de 06 de 2018). *Create Simple Deep Learning Network for Classification*. Fonte: MathWorks: https://www.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classification.html

Srivastava, N., Hilton, G., & Krizhevsky, A. (2014). Dropout: A Simple Way to Prevent Neural Networks from.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., & Reed, S. (2015). Going deeper with convolutions.

Transfer Learning Using GoogLeNet. (07 de 2018). Fonte: MathWorks: https://www.mathworks.com/help/nnet/examples/transfer-learning-using-googlenet.html