Reconhecimento de Modelos de Veículos

Fernando Benedito Veras Magalhães

January 15, 2018

1 Introdução

Em 2017, 2,1 milhões de automóveis, incluindo picapes e furgões, foram vendidos no Brasil. A variedade de modelos disponíveis ao consumidor tende a aumentar a medida que as montadores tentam produzir o carro perfeito para cada nicho do mercado. É comum que pessoas não saibam reconhecer os carros que observam no dia a dia, incluindo alguns dos modelos mais comuns. Um método de reconhecimento de veículos além de ajudar os curiosos, pode ajudar agentes de segurança a identificar automóveis cuja a placa não está visível ou pode ter sido adulterada.

O uso de aprendizado supervisionado de máquina pode levar ao desenvolvimento de um classificador eficiente capaz de predizer o modelo de um veículo tomando como entrada uma imagem do mesmo.

Esse trabalho buscar treinar redes neurais convolucionais para o reconhecimento do modelo de automóveis e possui como objetivos específicos:

- Pré-processar uma base de dados para o treinamento de redes neurais;
- Treinar diferentes arquiteturas de redes neurais;
- Comparar o desempenho de cada uma delas;

2 Fundamentação Teórica

2.1 Redes Neurais Convolucionais (CNNs)

CNNs são um tipo especial de redes neurais especialmente eficazes em extrair características de imagens, também são muito utilizadas para reconhecimento de fala e de linguagem natural. A utilização de CNNs permite diminuir o número de pesos em cada camada da rede além de representar melhor o relacionamento espacial entre pixels próximos. As CNNs buscam identificar padrões, sendo que suas camadas iniciais obtém características de baixo nível e alimentam as camadas posteriores obtendo características de cada vez mais alto nível. As CNNs são caracterizadas pelo uso das seguintes operações:

1. **Convolução:** A convolução consiste em obter neurônios que levam em consideração apenas uma região da imagem. Cada neurônio convolucional processa os dados referentes apenas a seu campo receptivo. Na figura 1 podemos notar como diferente das camadas de uma rede neural comum nas camadas convolucionais os neurônios não estão conectados a todos os neurônios da camada anterior.
2. **Pooling:** Consiste em combinar as características de píxels próximos tomando apenas a característica de maior valor (Max Pooling) ou a média delas (Average Pooling). Objetiva reduzir a dimensionalidade do espaço de características. A figura 2 demonstra a diferença entre Max Pooling e Average Pooling
3. **Relu:** Funciona como uma função identidade, porém retornando 0 para valores negativos.

2.2 GoogLeNet e o Módulo Inception

A CNN GoogLeNet foi vencedora de algumas categorias do *ImageNet Large-Scale Visual Recognition Challenge 2014*. O artigo [SLJ+15] propõe essa rede juntamente com seu módulo principal, o Inception.

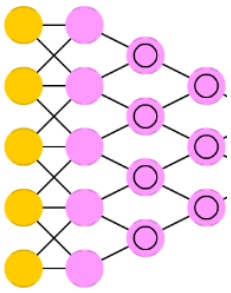


Figure 1: Algumas camadas convolucionais.

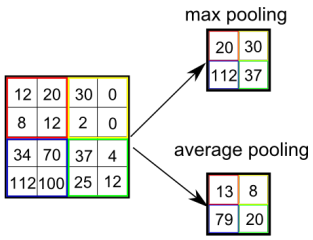


Figure 2: Max Pooling e Average Pooling.

2.2.1 Inception

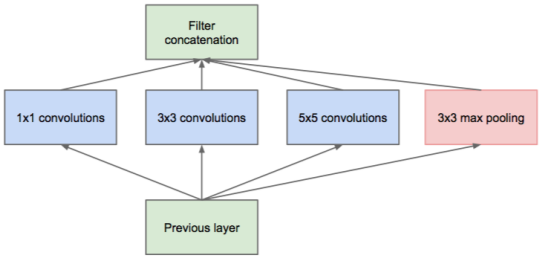
O módulo inception é caracterizado pela utilização filtros 1x1, 3x3 e 5x5 juntamente com um pooling os resultados desses filtros são combinados e enviados para a próxima camada. Dessa maneira, o módulo pode capturar características em diferente níveis e durante o treinamento, aquelas que forem mais relevantes irão se sobressair. A arquitetura do módulo inception dessa maneira está representada na figura 3(a). Porém essa arquitetura necessitaria de um grande número de pesos, o que tornaria a CNN mais custosa computacionalmente. Então decidiu-se adotar algumas convoluções 1x1 para reduzir a dimensionalidade dos filtros, obtendo como resultado a arquitetura da figura3(b). Assim, combinando módulos inception com redução de dimensionalidade, é possível construir redes profundas, capazes de obter características de alto nível e cada vez menos lineares, mas ainda assim necessitar de relativamente pouca memória e processamento para a inferência de uma classe.

2.2.2 GoogLeNet

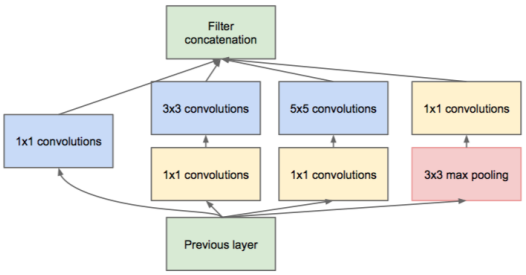
GoogLeNet foi a CNN baseada no módulo inception submetida para o ILSVRC 2014. A GoogLeNet começa com algumas convoluções e máx poolings e é seguida por 9 módulos inception. Possuindo ao final um Average Pooling, Uma camada Fully Conected e uma saída softmax. É interessante notar que a rede possui outras saídas no meio da arquitetura. Essas saídas secundárias são utilizadas apenas durante o treinamento e servem para garantir que os gradientes sejam propagados para as camadas longe da saída final da rede. A arquitetura da GoogLeNet está representada na figura 4

3 Dataset

O dataset [KSDFF13] utilizado é composto de 2120 imagens de 50 classes diferentes, o que inclui 10 marcas cada uma com 5 modelos o número de imagens por modelo é variado, ficando em torno de 37. Nas imagens, os veículos encontravam-se em posições variadas, incluindo fotos frontais, traseiras e várias fotos diagonais, as cameras e localidades onde as fotos foram tiradas também eram variadas. Originalmente, as imagens possuiam tamanhos variados e não estavam cortadas de modo a incluir apenas a bounding box dos veículos. A maioria das imagens estava colorida. O objetivo era usar como entrada imagens de tamanho 299,299 com 3 canais (RBG) para isso foi necessário pre-processar as imagens.



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figure 3: O módulo inception. Fonte:[SLJ+15]

3.1 Pré-processamento

O dataset incluia uma pasta com todas as imagens e um arquivo contendo a bounding box das imagens e a sua classe representada apenas por um número. Foi escrito um script para cortar as imagens de modo a conter apenas o bounding box e separá-las em pastas de acordo com a classe a qual pertenciam. Foi enviada uma mensagem para o email disponibilizado na página do dataset de modo a obter o modelo de veículo o qual cada número representava, mas nenhuma resposta foi obtida. Então foi necessário observar as imagens em busca dos modelos e por meio de pesquisas na internet foi possível obter o modelo que cada inteiro representava.
As imagens foram então divididas em 80% para treino e 20% para validação. Por fim foi necessário ajustar as imagens para o formato 299,299,3 e convertê-las para matrizes de modo que pudessem ser armazenadas em um único arquivo que pudesse ser lido na nuvem do google.

3.2 Aumento de dados

Devido ao número pequeno de imagens por veículo, foi utilizada uma técnica de aumento de dados nas imagens reservadas para treino, foram utilizados de forma aleatória durante o treinamento espelhamentos horizontais, inclinações de até 30 graus e zooms.

4 Treinamento

Foram adotadas 4 arquiteturas de redes neurais: uma CNN simples, uma com um módulo inception, uma com 3 módulos inceptions e por fim a GoogLeNet pré-inicializada com os pesos da base ImageNet. Seguem as arquiteturas das redes construídas utilizando a bibioteca keras [C+15]:

1. **SimpleConvNet:**
    - Convolução(32 filtros de (3,3), relu)
    - MaxPooling(32 filtros de (2,2))
    - Convolução(32 filtros de (3,3), relu)
    - MaxPooling(32 filtros de (2,2))
    - Convolução(64 filtros de (3,3), relu)
    - MaxPooling(64 filtros de (2,2))
    - Flaten()

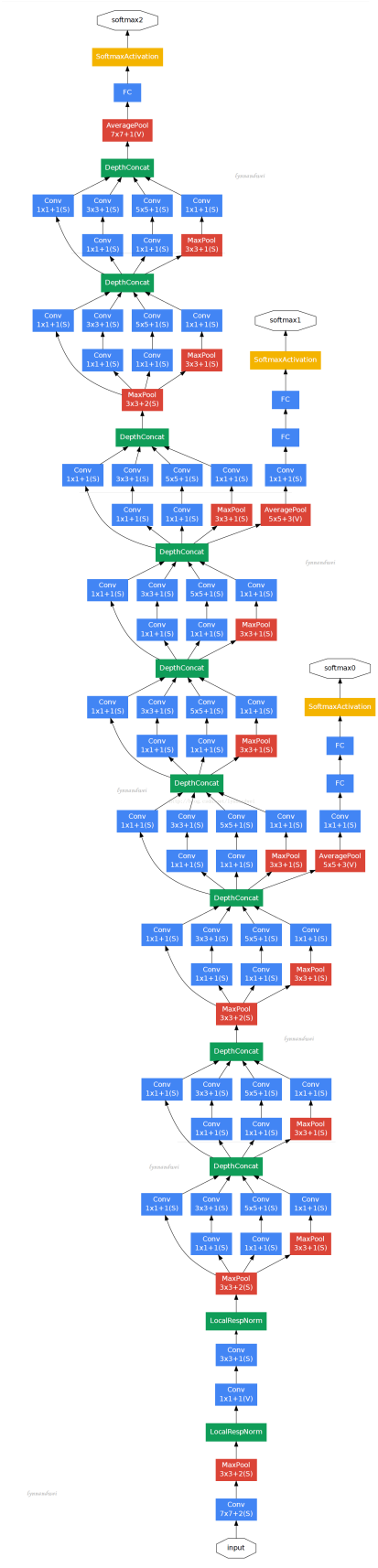


Figure 4: A rede GoogLeNet. Fonte:[SLJ+15]

- FullyConnected(128 saídas, relu)
- Dropout(50%)
- FullyConnected(50 saídas, relu)
- Sigmoid()

2. **SingleInception:**
   - Convolução(32 filtros de (3,3), stride(2,2) relu)
   - Convolução(32 filtros de (3,3), relu)
   - Convolução(64 filtros de (3,3), relu)
   - MaxPooling(64 filtros de (3,3), stride(2,2))
   - Convolução(80 filtros de (1,1), relu)
   - Convolução(192 filtros de (3,3), relu)
   - MaxPooling(192 filtros de (3,3), stride(2,2))
   - 1 Módulo inception
   - GlobalAveragePooling()
   - FullyConnected(50 saídas, relu)
   - Sigmoid()
3. **GoogLeNet:** Substitui o final da rede para retreinar de acordo com o novo dataset. Toma a saída do último módulo inception e adiciona:
   - GlobalAveragePooling()
   - FullyConnected(1024 saídas, relu)
   - FullyConnected(50 saídas, relu)
   - Sigmoid()

O treinamento foi feito em 100 épocas para todas as configurações de CNN, exceto para a GoogLeNet, cujo treinamento foi dividido em duas etapas:

- A primeira consistiu de 50 épocas treinando apenas as camadas que foram adicionadas ao final da rede.
- Na segunda essas últimas camadas foram treinadas juntamente com o último módulo inception da rede por mais 50 épocas.

O treinamento foi feito desse modo seguindo o tutorial de *fine tune* apresentado em: https://keras.io/applications/#inceptionv3

5 Resultados

A SimpleConvNet apresentou acurácia de 5% no dataset de treino e 6% no dataset de validação(figuras 5 e 6). A rede com 1 módulo inception apresentou uma acurácia de 93% no dataset de treino e 49% no dataset de validação(figuras 7 e 8).

Ao final das primeirs 50 épocas, a GoogLeNet apresentou uma acurácia de 63% no dataset de treino e 48% no dataset de validação (figuras 9 e 10). Treinadas mais 50 épocas seguindo o descrito na seção anterior, a acurácia foi de 96% no dataset de treino e 67% no de validação(figuras 11 e 12).

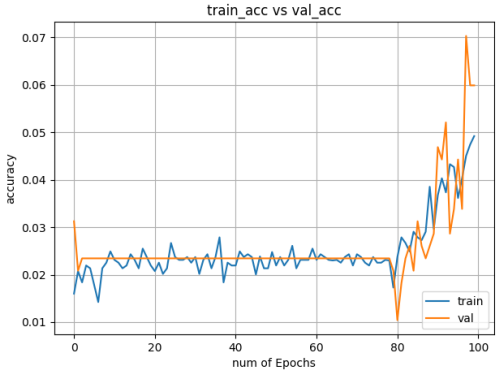


Figure 5: Acurácia SimpleConvNet.

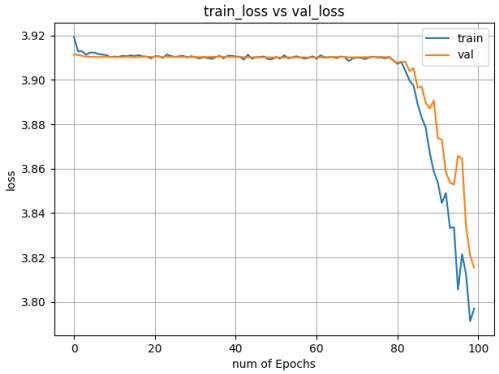


Figure 6: Loss SimpleConvNet.

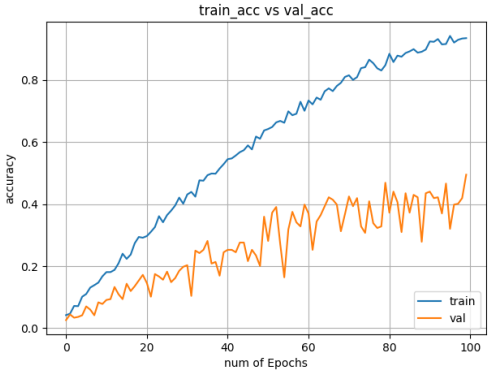


Figure 7: Acurácia Single Inception.

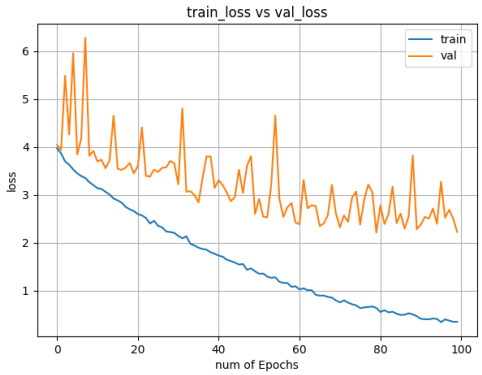


Figure 8: Loss Single Inception.

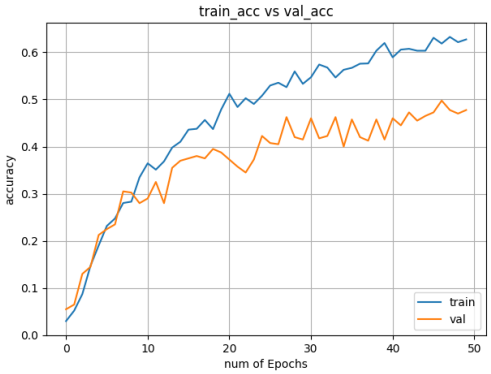


Figure 9: Acurácia GoogLeNet etapa 1.

Figure 10: Loss GoogLeNet etapa 1.

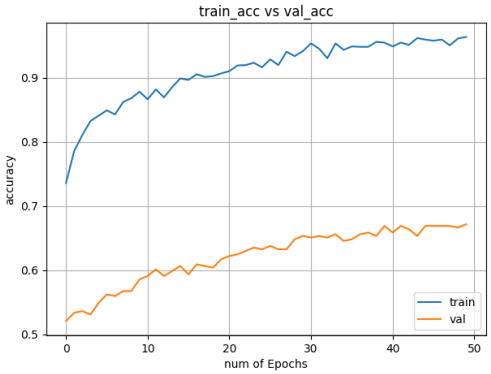


Figure 11: Acurácia GoogLeNet etapa 2.

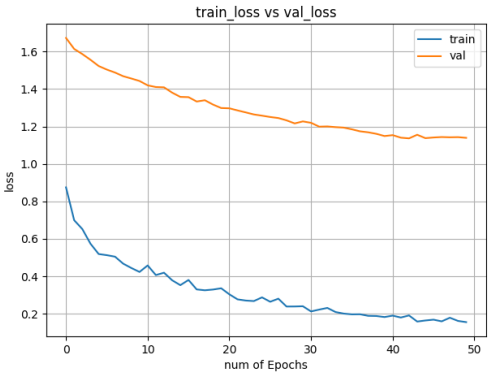


Figure 12: Loss GoogLeNet etapa 2.

6 Conclusão

Podemos notar que uma rede muito simples como a SimpleConvNet descrita é completamente incapaz de resolver um problema tão complexo quanto o proposto. Já o uso de uma rede com algumas camadas convolucionais e de pooling seguida de um módulo inception apresentou um desempenho muito bom, 93% de acurácia no dataset de treino, porém notamos que a rede pode sofrer com overfitting devido ao desempenho ruim no dataset de validação. Ao utilizar uma rede conhecida e com bons resultados no ImageNet como a GoogleNet notamos que ela apresentou um desempenho ainda melhor no dataset de treino e sofreu menos com o possível overfitting, 67% de acurácia no dataset de validação. Não pode-se concluir que há um grande overfitting nas redes devido ao dataset conter poucas imagens por veículo e alta variação entre elas que acontece pois as fotos foram tiradas de várias posições diferentes. Mesmo utilizando técnicas de aumento de dados, alguns veículos possuem apenas 3 fotos que mostram a traseira. Porém devido a alta acurácia no treinamento das duas últimas redes em apenas 100 épocas, pode-se ter alguma confiança de que o problema possa ser resolvido utilizando um dataset com mais imagens por veículo ou fotos que foram tiradas de posições similares.

References

[C+15] François Chollet et al. Keras, 2015.

[KSDFF13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.