Tópicos Especiais em Processamento Digital de Imagens:

Relatório do Segundo Trabalho Prático

Luiz Fernando Fonseca Pinheiro de Lima

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2019

1 INTRODUÇÃO

Este é o relatório para o segundo trabalho prático da disciplina **Tópicos Especiais em Processamento Digital de Imagens**, na qual é abordado assuntos sobre redes neurais e, mais especificamente, redes neurais convolucionais.

1.1 Definição do Trabalho

No segundo trabalho da disciplina, deveríamos implementar a arquitetura MobileNet e trabalhar com a base de dados Tiny ImageNet.

1.2 Objetivo Geral

Foram objetivos desse trabalho:

- implementar a arquitetura MobileNet;
- por em prática conceitos de separação de base de dados, definição de hiperparâmetros, treino, validação e teste de uma rede.

1.3 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma: A seção 2 traz uma breve explicação sobre a arquitetura de rede convolucional implementada, a MobileNet e sobre a base de dados Tiny ImageNet. A seção 3 explica a metodologia do trabalho, trazendo as ferramentas utilizadas, importação e ajustes da base de dados e definição inicial de alguns hiperparâmetros. Na seção 4 são apresentados os testes feitos com as mudanças dos hiperparâmetros, seus resultados e as análises feitas a partir deles. Por fim, a seção 5 apresenta a conclusão do trabalho.

2 CONCEITOS GERAIS

A seção 2 traz uma breve explicação sobre a arquitetura de rede convolucional MobileNet e sobre a base de dados Tiny ImageNet, usadas nesse trabalho.

2.1 MobileNet

Além das camadas tradicionais de convolução, a arquitetura MobileNet utiliza o conceito de de convolução separável em profundidade, do inglês, depthwise separable convolution, para buscar reduzir a complexidade das operações de convolução (Figura 1).

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figura 1: Arquitetura MobileNet

A entrada da MobileNet é uma imagem RGB, com dimensões 3@224x224 pixels, que passa por uma sequência de camadas convolucionais e camadas convolucionais separáveis em profundidade.

Essa arquitetura não utiliza camadas de pooling para reduzir as dimensões das imagens. Essa redução é feita em algumas camadas de convolução, nas quais se utiliza stride igual a dois, ou seja, a imagem é reduzida pela metade.

A única camada de pooling presente na MobileNet se encontra entre a última camada convolucional e as duas últimas camadas ultilizadas para a classificação, com filtro de tamanho 7x7.

A penúltima camada é uma camada flatten, que formata os valores para a quantidade de neurônios necessários para na saída.

Por fim, a camada de saída, que é uma camada densa com 1000 neurônios, onde o valor de saída de cada um deles representa os valores de confiança para cada uma das classes da base de dados, essa camada utiliza como função de ativação a função softmax.

2.2 Tiny ImageNet

O Tiny ImageNet é uma base de imagens subconjunto da base ImageNet, apresenta imagens RGB com dimensões 64x64 de 200 classes e que conta com 110000 registros rotulados.

3 METODOLOGIA

Para o desenvolvimento desse trabalho foram utilizadas as seguintes ferramentas:

- Linguagem de programação Python, versão 3.5.2, para a implementação da rede;
- Bilioteca Keras, versão 2.2.4, como interface para desenvolver a rede convolucional;
- Biblioteca TensorFlow, versão 1.11.0, como backend do Keras;
- Biblioteca numérica NumPy, versão 1.14.2, para manipulação de arrays, vetores e matrizes;
- Biblioteca Matplotlib, versão 2.2.2, para visualização de gráficos;
- Biblioteca OpenCV, versão 3.4.2, para manipulação de imagens.

O trabalho foi realizado em um computador com sistema operacional Linux 16.04, CPU Intel i3-4005U quad-core com clock 1.70GHz e 8GB de memória RAM.

3.1 Importando e Ajustando o Tiny ImageNet

A partir do link compartilhado da base de dados, a separação das imagens de treino, validação e testes foi realizada utilizando o algoritmo do aluno Gabriel Leite Santana. Ficando 90000 imagens para treinamento, 10000 para validação e 10000 para teste.

As imagens não foram redimensionadas para o tamanho utilizado originalmente na arquitetura MobileNet.

3.2 Implementação Inicial

A implementação da rede seguiu todas as camadas e número de filtros e apresentada na arquitetura MobileNet da figura 1.

Os hiperparâmetros, inicialmente, foram definidos a partir do estado da arte do turma:

- Funções de ativação: a relu foi utilizada nas camadas convolucionais, bem como nas duas primeiras camadas densas, já na última camada densa foi utilizada a função softmax;
- Otimizador: o Adam foi utilizado como otimizador para a rede, com learning rate = 5e-4 e decay = 0.0;

- Inicialização dos pesos: o algoritmo he_normal foi utilizado como inicialização dos pesos;
- Função de perda: a entropia cruzada foi utilizada como função de perda para a rede.

Para visualizar os resultados de acurácia de treino e validação a partir da época 0, foi usado o parâmetro **initial_epoch()**, também definido na função **fit()**, com o valor -1.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Esta seção descreve o teste realizado no trabalho.

O teste utilizou 20 épocas e batch de 32 imagens para o treinamento, também definidos pelo estado da arte da turma.

A rede atingiu as acurácias de 69.69% para o conjunto de treinamento e de 27.16% para o conjunto de validação e suas curvas podem ser vistas na figura 2.

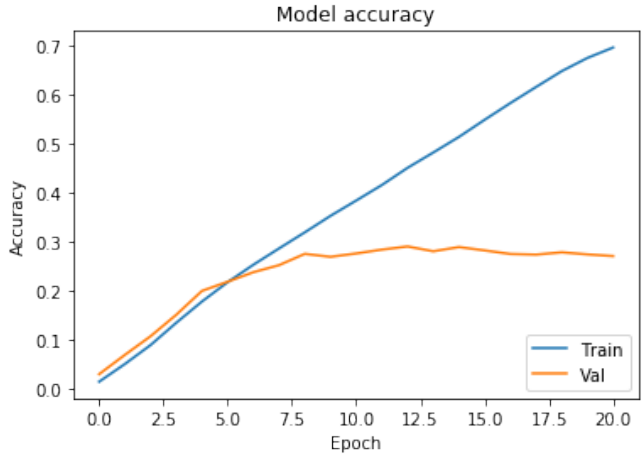

Figura 2: Curvas de acurácias de treino e validação para o teste

Assim como para o conjunto de validação, para a base de teste a rede obteve uma baixa acurácia, pontuando apenas 26.68%.

Possivelmente, a rede se ajustou muito para as imagens de treino, não conseguindo classificar corretamente imagens de uma classe com mais diferenças daquelas que foram apresentadas no treinamento.

Para tentar sanar esse problema, poderia ser feita uma alteração na redução da quantidade de strides que redizem as dimensões das imagens. Mas, por limitações de hardware, o treinamento ficaria inviável para a entrega desse trabalho, já que cada época do treinamento iria ser executada em, aproximadamente, 9 horas.

5 CONCLUSÕES

Por limitações de hardware, apenas um teste pôde ser realizado.

Para tentar contornar esse problema, a ferramenta Collab da Google foi estudada para ser utilizada, mas devido ao tempo para a entrega desse trabalho, ficou inviável destinar esforço para acompanhar a curva de aprendizado da ferramenta.

Entretanto, o objetivo principal do trabalho foi alcançado, uma vez que a arquitetura de rede neural convolucional MobileNet foi implementada. E ainda, esse trabalho serviu para que as limitações de hardware para as redes neurais convolucionais pudessem ser observadas.