Aprendizagem Profunda

Abdullah Zaiter - 15/0089392

Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília

***Abstract*— This document presents the results of the third exercise (Deep Learning) of the discipline Introduction to Numerical Intelligent Control at the 2/2018 semester**

***Keywords*— Neural Networks, caltech101, convolutional networks, systems identification.**

***Resumo*— Este documento apresenta os resultados obtidos no terceiro exercício (Aprendizagem Profunda) da disciplina Introdução ao Controle Inteligente Numérico no semestre 2/2018 .**

***Palavras chave*— Redes neurais, caltech101, Redes convolucionais, Aprendizagem Profunda.**

I. Objetivos

Utilizando o banco de dados CalTech-101, construir uma rede neural profunda para reconhecer objetos em imagens digitais.

II. Ambiente e Dados utilizados

A Máquina utilizada possui placa de video dedicada Nvidia 1070 - 8 GB[1], a mesma foi utilizada para o treinamento e validação das redes neurais estudadas neste exercício.
A decisão do uso da própria GPU e não a DevCloud para realizar o treinamento é devido ao tempo parecido entre os dois operando sobre o backend Tensorflow, não foi obtido um desempenho mais alto no DevCloud pois o mesmo não é otimizado para operar com o backend Tensorflow e sim com o próprio Backend da Intel, que no caso se chama Neon.

III. Metodologia de testes

Utilizou-se o código passado pelo professor[4] e analisou-se o git[5] de origem e foram realizadas algumas alterações variando algumas característica de cada rede.
Para entender melhor o funcionamento de aprendizagem profunda e de redes convolucionais, foram treinadas cinco redes neurais com topologias variadas, além disso, para validar oque foi estudado em teoria, foi variado também o tamanho do Kernel em algumas cámadas.

Encontram-se na tabela a seguir os treinamentos que foram realizados

Rede Nº	Qt. Camadas	Kernel	Taxa de aprendizagem
1	28	3x3	0.0001
2	30	3x3	0.0001
3	27	3x3	0.001
4	31	3x3	0.0001
5	31	6x6	0.0001

Tabela 1. Redes treinadas

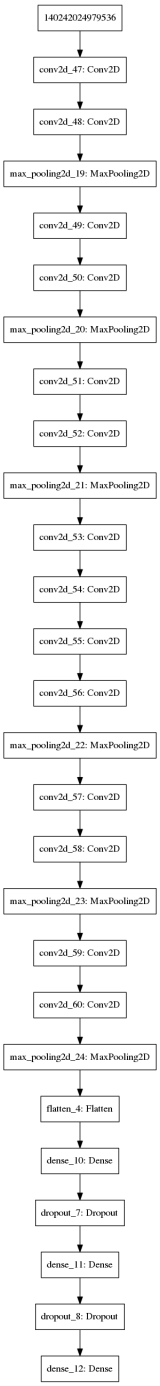

Fig. 1. Topologia da rede número 3.

Não foi possível fazer um tratamento de imagens melhor como duplicar as fotos das categorias que tem quantidade

menor de fotos, as fotos são carregadas pela maneira que o professor passou o código, pegando uma amostra aleatória de 10% do total. Junto com este relatório está as fotos das topologias de todas as redes treinadas.

IV. Dados e Resultados Obtidos

As redes treinadas com os resultados respectivamente

Rede Nº	Qt. Camadas	Kernel	Acerto
1	28	3x3	52%
2	30	3x3	48%
3	27	3x3	50%
4	31	3x3	44%
5	31	6x6	40%

Tabela 2. Redes treinadas e resultados

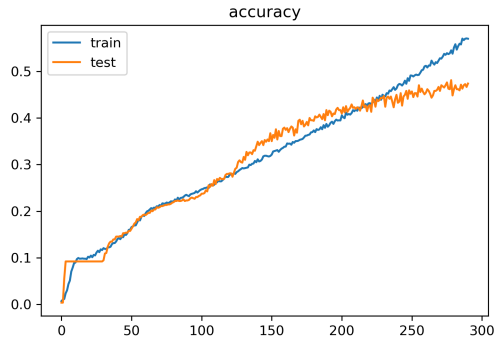

Fig. 2. Acurácia para treinamento e validação da rede numero 2.

Mudando o tamanho do kernel entre a rede 4 e a rede 5 fez com que a acurácia na validação caísse em 4%.

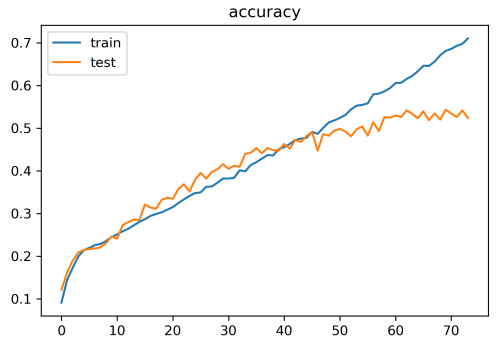

Fig. 3. Acurácia para treinamento e validação da rede numero 5.

Junto com o relatorio, será enviado o modelo Json de todas as redes treinadas.

V. Conclusões

Concluiu-se que o número de camadas interfere diretamente no resultado da rede e que esta relação não necessariamente é proporcional, logo, maior número de camadas não necessariamente resulta em resultados mais exatos. Bem como o tamanho do Kernel, um kernel maior apresentou um resultado pior para a mesma topologia da rede. Uma taxa de aprendizagem mais alta (usada na rede número 3) fez com que tenha taxa de acerto mais alta nos dados de treinamento mas essa taxa caiu nos dados de validação de uma forma maior que as outras redes, logo, tem que ser escolhida uma taxa de aprendizagem descente para cada aplicação. [**?**][**?**]

Referencias

[1] Geforce-Gtx-1070-ti<https://www.nvidia.com/pt-br/geforce/products/10series/geforce-gtx-1070-ti>. Acesso em: 28 out. 2018.

[2] Tutorial para utilização da placa de video com o tensorflow<https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/tensorflow/>. Acesso em: 28 out. 2018.

[3] Data Science – Regressões com Python<https://imasters.com.br/back-end/data-science-regressoes-com-python>. Acesso em: 23 sep. 2018.

[4] Arquivo de Python disponibilizado pelo professor<http://www2.ene.unb.br/adolfo/Lectures/ICIN/Ex3_Python.zip>. Acesso em: 28 out. 2018.

[5] Repositório de códigos uteis<https://github.com/acht7111020/CNN_object_classification>. Acesso em: 28 out. 2018.