Summer ML Triggers

Week 1 Report

Russell Zhang
dg22882@bristol.ac.uk

Supervisor Dr Sudarshan Paramesvaran

Particle Physics Laboratory

Tables

Part 1

Paper Reading

Neural network triggers for the CMS detector at the HL-LHC

Content:

- I focused on reading the section on machine learning and neural networks. This paper establishes a CNN architecture with two layers of convolutional layers and two layers of fully connected neural networks.
- I read the introductory section on particle physics and built up an initial understanding.

Overview of ML part

Introduction

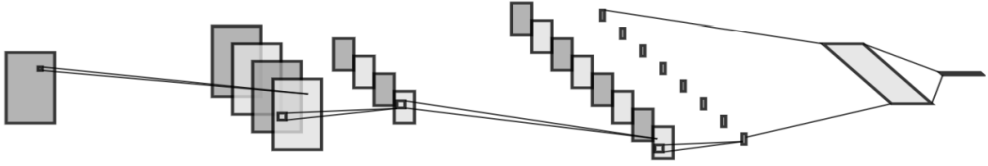
The LHC can generate hundreds of millions of data in microseconds. But physicists are interested in only some of them. Research aims to filter out data of interest using neural networks. At present, the signal data $HH \to b\bar{b}b\bar{b}$ collected by the LHC collector can be pre-processing in 2D image format, so we can apply convolutional neural network for deep learning. The goal is to compare the performance of different architectures of neural networks to selecting signals.

Feed-Forward Neural networks Algorithm

$$z_j^{(1)} = h(\sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) \tag{1}$$

Neural Networks

This is the neural network architecture constructed in the paper.



Future Work

- I plan to read Chapter 1 of The Phase-2 Upgrade of the CMS Level-1 Trigger in the second week to deepen my understanding of the physics background. I think this is important because the research work will investigate Graph Neural Networks in the next following weeks. The input data format of GNN is different from that of CNN. Research needs to remodel raw data, so I need a clear understanding of the physical definition.

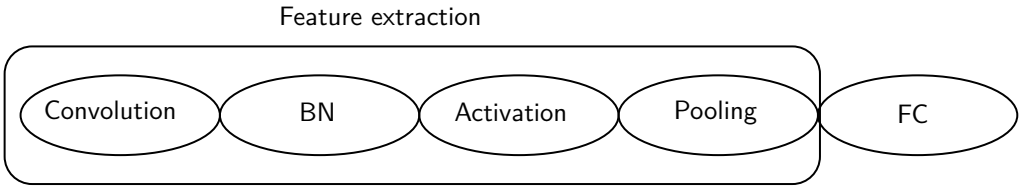
Part 2

Convolutional Neural Networks

Content:

- This is a part of my work at the beginning of week 1. I used to study classic artificial neural networks. So, I reviewed the mathematical foundations and inference of neural networks and have started writing a detailed note by hand.
- After the recall, I learned the basic convolutional neural networks and built a CNN architecture with one convolutional layer and a two-layer fully connected neural network. Then, I successfully implemented it in Python. I did ten rounds of training in total. The improvements in the model training were more obvious in the first five rounds of experiments, and after more rounds of experiments, the training encountered barriers. Mostly because the neural network architecture is too simple.
- I tried a new CNN architecture—ResNet (code from Internet but suitable for same datasets) in the next part and made great progress.

Common Structure Revise



BN: Batch Normalization
FC: Full-connected neural networks

I constructed a baseline convolutional neural networks and performed ten rounds of training experiments.

Inference

Inference for Feed-Forward Neural networks Algorithm

Set up a one-layer Neural Network,

Input $\rightarrow 1^{st}$ Neural Layer: $z_j^{(1)} = h(\sum_{i=1}^{N} w_{ji} x_i + w_{j0})$

1^{st} Neural Layer $\rightarrow$ Output: $y_k(\mathbf{x}, \mathbf{w}) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(1)}))$

Appendix:

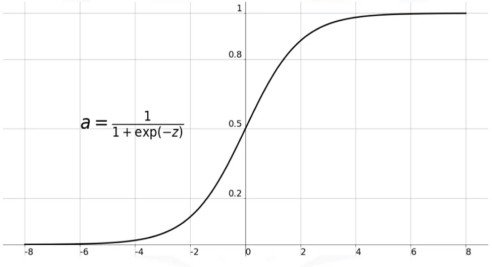
Sigmoid function: $\sigma(\alpha) = \frac{1}{1+exp(-\alpha)}$

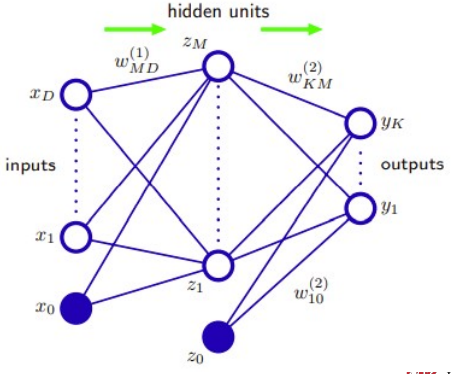
(This is some parts from my handwritten notes and I will put full version on my Github and Upgrade every week.)

Inference

Graph for Inference

Sigmoid Function

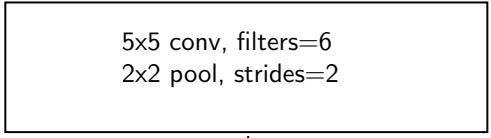




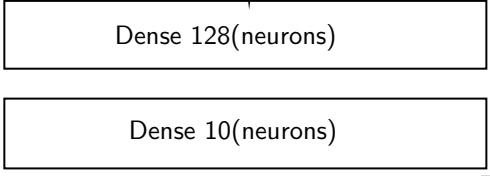
Architecture

One convolutional layer
Two full connected feed-forward neural networks

Convolutional Layer



FC



Basic Code

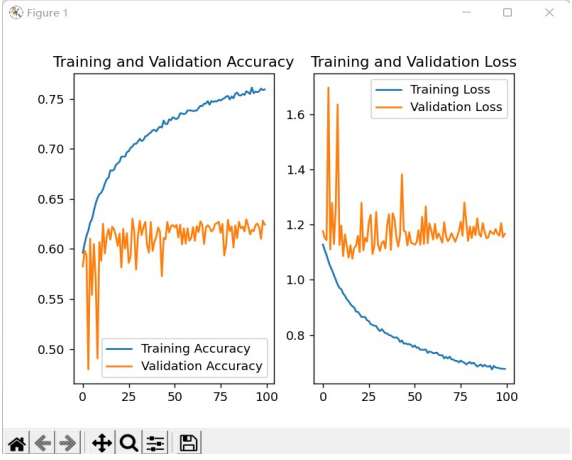
(Take dataset cifar10 for example)

```
    # Convolutional Layer
    Conv2D(filters=6, kernel_size=(5, 5), padding='same')
    #BatchNormalization
    BatchNormalization()
    #Activation
    Activation('relu')
    #Pooling
    MaxPool2D(pool_size=(2, 2), strides=2, padding='same')
    #Dropout
    Dropout(0.2)
```

1st ACC and loss plot with 5 epochs

- The model accuracy is low and the loss is very high, but the fit is good.
- The number of times the model was trained is small, and we cannot confirm whether it is suitable or not.

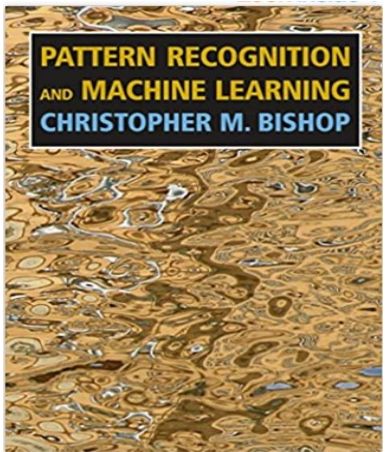
10th training: ACC and loss plot with 100 epochs



After the model has been trained a lot, the loss and accuracy curves of the testing set tend to be stable. Experiments run into barriers. The main possible problem is that the NN model architecture is simple, or the local minima trap is encountered during the gradient descent process.

Reference

- My handwritten notes for neural networks are mainly based on *Bishop C M, Nasrabadi N M. Pattern recognition and machine learning[M]. New York: springer, 2006.*



Part 3

Potential Research Directions

There were two potential research topics which I researched in week 1.

- Advance architecture of CNN
- Bayesian convolutional neural networks

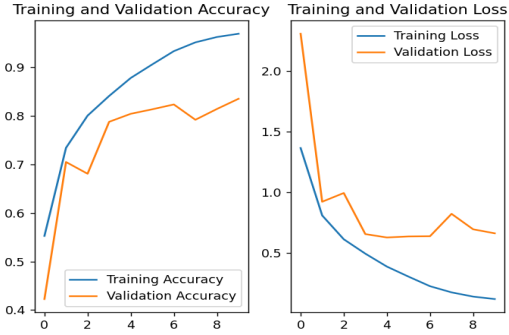
But after meeting on July 15th, with the advisor's suggestion, these two directions did not correspond to the main problems of the project (or have too little impact). I think I need to focus on finding the optimal neural network for selecting signals.

Overview

Advance architecture of CNN

1. LeNet 1998
   Reduce coeficients
2. AlexNet 2012
   Apply ReLu activation function to boost training speed
   Apply Dropout to alleviate overfitting
3. VGGNet 2014
4. Inception Net 2014
5. ResNet 2015

Experiment of ResNet



Compared with the baseline model in the previous part, the experimental results achieve a significant improvement. This is the result of the first experiment, and only ten epochs of the model were trained. The loss of the training set and testing set is highly fitted and tends to be less than 0.5. The training set accuracy is close to 1.0, and the test set accuracy exceeds 0.8.

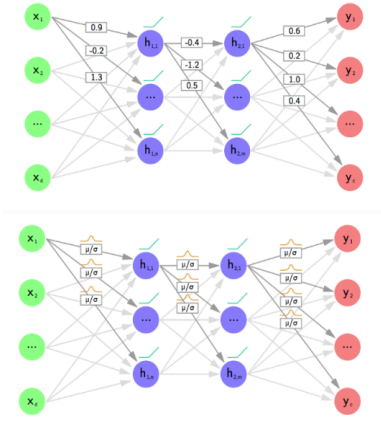
Overview

abstract

Convolutional neural networks (CNNs) work well on large datasets. But labelled data is hard to collect, and in some applications larger amounts of data are not available. The problem then is how to use CNNs with small data – as CNNs overfit quickly. We present an efficient Bayesian CNN, offering better robustness to over-fitting on small data than traditional approaches. This is by placing a probability distribution over the CNN's kernels. We approximate our model's intractable posterior with Bernoulli variational distributions, requiring no additional model parameters.

On the theoretical side, we cast dropout network training as approximate inference in Bayesian neural networks. This allows us to implement our model using existing tools in deep learning with no increase in time complexity, while highlighting a negative result in the field. We show a considerable improvement in classification accuracy compared to standard techniques and improve on published state-of-the-art results for CIFAR-10.

Structure of BayesCNN



- Top: Each filter weight has a fixed value, as in the case of frequentist Convolutional Networks.
- Bottom: Each filter weight has a distribution, as in the case of Bayesian Convolutional Networks.

Keywords

- 1 Convolutional neural networks (CNNs) work well on large datasets. But labelled data is hard to collect, and in some applications larger amounts of data are not available. The problem then is how to use CNNs with small data – as CNNs overfit quickly. We present an efficient Bayesian CNN, offering better robustness to over-fitting on small data than traditional approaches.
- 2 Due to the capacity of CNNs to fit on a wide diversity of non-linear data points, they require a large amount of training data. This often makes CNNs and Neural Networks in general, prone to overfitting on small datasets. The model tends to fit well to the training data, but are not predictive for new data. This often makes the Neural Networks incapable of correctly assessing the uncertainty in the training data and hence leads to overly confident decisions about the correct class, prediction or action.

Limitation

- The problems solved by Bayesian convolutional deep networks are limited to regularization and overfitting. This is not our main research question.

Reference

1

Gal Y, Ghahramani Z. Bayesian convolutional neural networks with Bernoulli approximate variational inference[J]. arXiv preprint arXiv:1506.02158, 2015.

2

Shridhar K, Laumann F, Maurin A L, et al. Bayesian convolutional neural networks[J]. arXiv preprint arXiv:1806.05978, 2018.

3

Shridhar K, Laumann F, Liwicki M. A comprehensive guide to bayesian convolutional neural network with variational inference[J]. arXiv preprint arXiv:1901.02731, 2019.

Additional Work

- Link to eduroam network
- Link to Linux Server (by VS Code)
- Set up a repository for recording project progress on Github

Plans for Week 2

- Start studying on Graph Neural Networks
- Keep studying on Convolutional Neural Networks
- Learn how to use Linux Server
- Learn resources on GitLab
- Implement CNN architecture in the paper for real datasets
- More based on supervisor's guidance and research process

Thanks