Machine Learning Research Engineer Test Task: From Idea to Implementation

Hasan Sait Arslan

Abstract

This documentation presents an overview of the practical coding task given for the Machine Learning Research Engineer interview test. The task documentation refers to a baseline implementation of Spatial Transformer Networks (STN) for solving the MNIST image classification problem. In this task, we use "CoordConv" layers replaced with the vanilla Convolutional layers at the first step. Secondly, using the vanilla convolutional layers, we utilize the state-of-the-art Multi-Head Attention layers instead of Spatial Transformer Networks. The architecture of STN with "CoordConv" does not demonstrate any drop in the validation loss and increase of accuracy. On the other hand, we observe a reduction in the validation loss and equivalent accuracy with the second experiment. Additionally, the architecture with the Multi-Head Attention has the least number of parameters among three architectures.

1 Introduction

The given practical coding task has three main steps:

1. Using the baseline implementation[1] for training the baseline Convolutional Neural Networks (CNN) [Fuk88] powered model with Spatial Transformer Networks (STN) [JSZ$^+$15] for the MNIST [LBBH98] image classification task, and observing the results on the MNIST test set.

2. Utilizing the CoordConv [LLM$^+$18] variety of convolutional layers instead of the vanilla settings on the baseline network, training the outcome model on the same dataset, and evaluating the results on the test set with the comparison of the baseline.

3. Testing the usage of Multi-Head-Attention Layers [VSP$^+$17] in the baseline setting instead of STNs and discuss the results with the comparison of the baseline and **CoordConv** setting.

You can see the source of the coding task from the github repository[2].

[1]https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html
[2]https://github.com/saitarslanboun/Veriff_Task

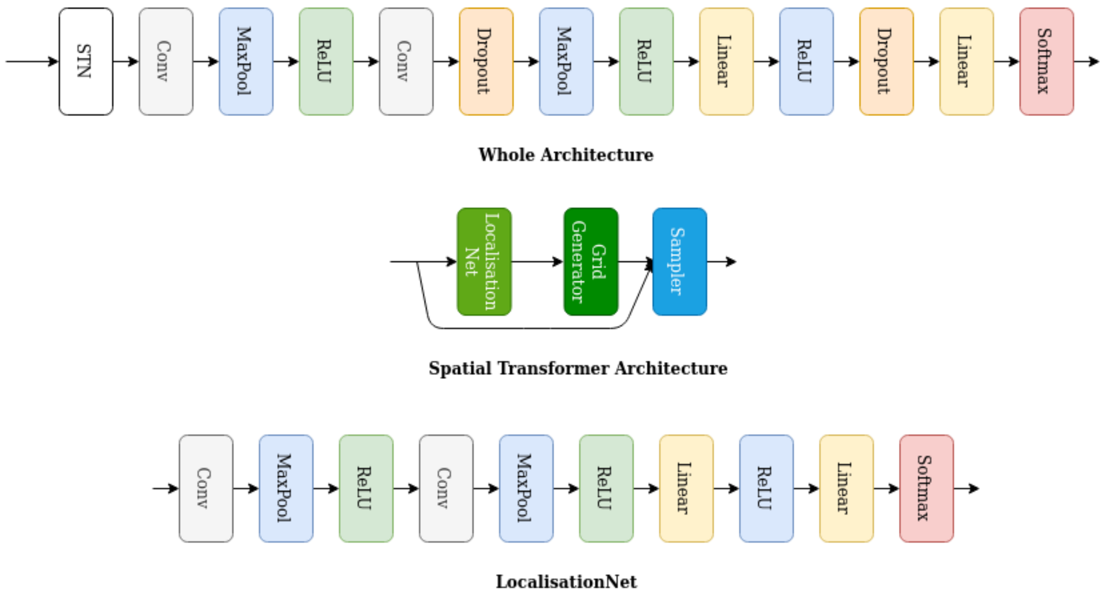

Figure 1: Baseline architecture with Spatial Transformer Network

2 Model Architectures

In this section, we demonstrate the architectural overviews of three different settings used in three expected experiments.

2.1 Spatial Transfomer Architecture Baseline

The tutorial page has a brief overview of the STN baseline architecture, and we demonstrate the whole image classification network in Figure 1. The output of the STN layer is fed through several convolutional and feedforward layers for the final classification. As it was suggested to use the implementation code on the tutorial page as the baseline, we use the whole architecture without any minor changes.

2.2 Spatial Transformer Architecture with CoordConv

In this setting, we replace all the convolutional operations with the CoordConv variety introduced in [LLM+18]. The vanilla CoordConv was used only on the first convolutional layer in the introductory CoordConv work. In our work, we use it for all the convolutional operations as suggested in the task guidelines.

2.3 Multi-Head Attention Supported Classification

In this setting (See Figure 2), we omit STN and employ Multi-Head Attention layers after each activation function (ReLU) of the two convolutional layers. The output channel sizes of the convolutional

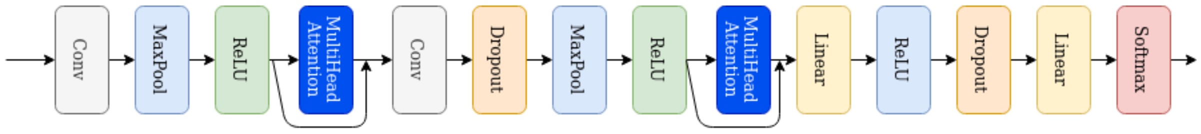

Figure 2: Architecture with MultiHead Attention

layers are 10 and 20, respectively. This attention layer was first introduced in [VSP+17] and used for Neural Machine Translation (NMT) and the Neural Machine Transliteration tasks. The multi-Head attention layer computes attention separately for the separate parts of the input embeddings. The authors found it beneficial to have multiple attention weights for a single input embedding rather than having a single attention weight as in the vanilla implementation [BCB14]. While the vanilla Multi-Head Attention layer has eight heads in the original implementation, we use two heads due to the narrow channel size of the convolutional layers. Having narrower embeddings would reduce the generalization capacity of the network.

While this attention layer was firstly used for Natural Language Processing (NLP) tasks, later it was introduced to be used in the Computer Vision (CV) tasks [PVU+18, BZV+19, DBK+20].

STN augments the input images by applying spatial attention. With Multi-Head attention, the input image is augmented with the residual connection of the attention weights spanning all grids of the image. STN is a single network that augments the input images initially, and the augmented images are used as the input for the classifier. Multi-Head Attention layers augment the gradients of the convolutional outputs within the classifier architecture.

3 Experiments and Results

In this section, we briefly represent the details of three different experiments that were carried out for the practical coding task.

3.1 Dataset

For all the experiments, we use the MNIST dataset, identical to the baseline tutorial. The dataset consists of 60K training and 10K testing samples. We use the testing data as the validation set during the training.

3.2 Training

We have the identical training setting of the baseline tutorial.

The training images are normalized with the mean and standard deviation of 0.1307 and 0.3081, respectively. At every epoch, we shuffle the training samples and use 64-minibatch sizes of input per training iteration.

We utilize the Stochastic Gradient Descent [RM51] optimizer with the learning rate of 0.01, and the Cross Entropy Loss function to calculate the loss.

The authors of the tutorial implementation employ Dropout with the rate of 0.5 after the second convolutional and the second feed-forward layers of the classifier. It means the gradients are randomly cut with the ratio of 0.5. While Dropout is a practical solution for the overfitting problem, the ratio parameter should be studied for different architectural settings to get the best validation accuracy. We do not experiment on this hyperparameter within the scope of our task.

We train all the experiments with 300 epochs without early-stopping. For each setting, using different **early-stopping** and **epoch** hyperparameters could improve the final accuracy and reduce the final validation loss. However, we do not study these parameters in our coding task. We record the epoch-wise validation loss and accuracy, and evaluate the architectures according to the behavior they have shown during the training.

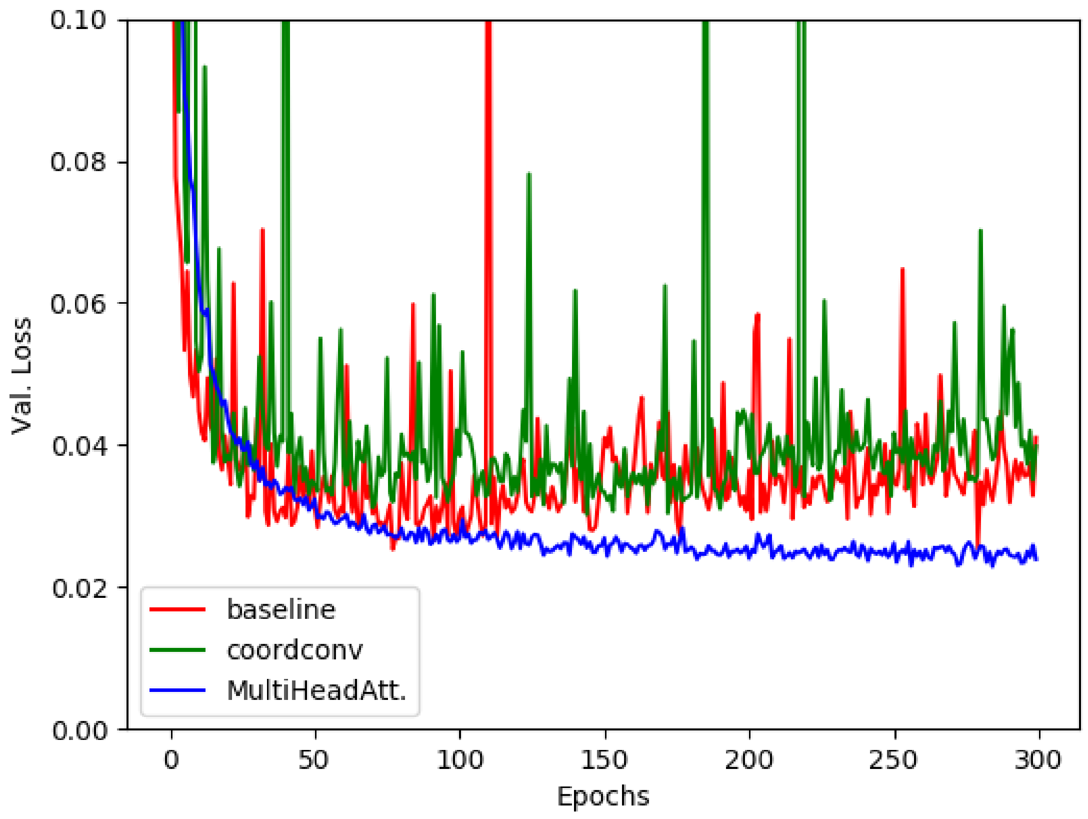

Figure 3: Validation loss change for three different experiments

3.3 Results

We observe the highest accuracy of 99.32%, 99.31%, 99.34% from the baseline STN, STN with CoordConv, and the classifier with the Multi-Head attention models, respectively. All of the models demonstrate close to the perfect accuracy (100%), and there is no significant difference of accuracies.

However, as (Figure 3) represents, the classifier with the Multi-Head Attention setting demonstrates a stable training without fluctuations between different epochs. All architectures consist of identical dropout layers. It brings the ratio of 0.5 randomness of activations. Multi-Head Attention brings a significant generalization ability to the activation gradients as we do not see a jump of loss values between epochs.

The baseline model has **27,360** parameters, CoordConv+STN has **37,384** and the classifier with Multi-Head Attention model has **25,460** parameters. We catch the most optimum performance with the least number of parameters within the experimental setting.

4 Conclusion and Suggestions

In this tutorial of the practical coding task, we have represented the brief documentation of three experiments, an experiment of baseline MNIST classifier with STN, replacing CoordConv with vanilla convolutional layers of the baseline setting, and the novel classifier with Multi-Head Attention layers. We have observed the most stable training schedule by obtaining the lowest validation loss while having the lowest number of architectural parameters.

It would be reasonable to carry additional experiments on the usage of dropouts, the learning rate scheduler, and the optimizer, as they might affect the training performance and the maximum validation accuracy. However, since it is out of the scope for this task, we skip that part for the moment.

References

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[BZV+19] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Fuk88] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[JSZ$^+$15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LLM$^+$18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018.

[PVU$^+$18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.