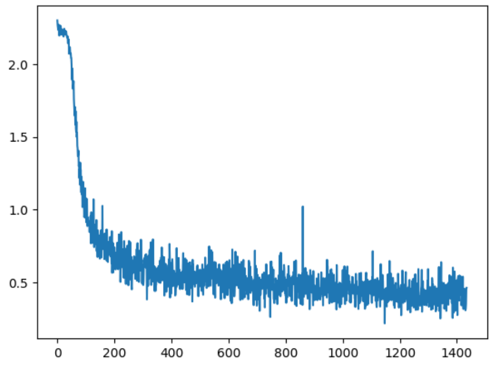
Lab 3 – Convolutional Neural Nets

1. Hyperparameter Optimization:

In this section we will explore several changes in the models architecture and training we take in order to find an optimal configuration to train on the SVHM dataset, which consists of RGB images with a 1-digit number at the centre of it with its corresponding label.

The initial configuration we explore is that of a 4 layer network that receives a $32x32x3$ image which is initially convoluted with a 16 channel $5x5$ **convolutional filter** adding 2 pixels of padding. Due to the fact that a $5x5$ filter reduces by 2 pixels the height and weight dimensions, and the padding being of 2, the format of the dimensions of the layer are $32x32x16$ which is then max pooled into $16x16x16$. After that, the same happens, this time with a $3x3$ **filter** of 32 channels with a padding of 1, and a max poolings, resulting in an output of $8x8x32$. Finally, this is connected to a linear **fully connected layer** of $8 \cdot 8 \cdot 32 = 2048$ nodes, that are then connected to a 10-node **output layer**.

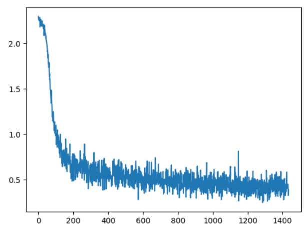
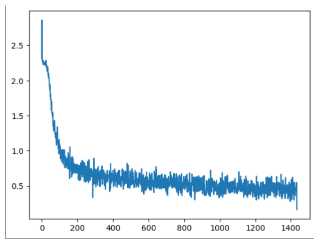
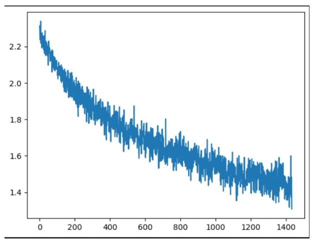


The result of this base model after 5 epochs and a batch size of 128, trained under the cross entropy loss and optimised through Adam, is of 86.28, measured as the accuracy of the model on the testing data.

1.1 Network Depth

The first thing we try is to add additional convolutional layers and change kernel sizes. In our case, we implement the same logic as before (conv+max followed by fully connected layers). Our only change with respect to the base model is that we added a convolutional layer with a $5x5$ kernel in between the $5x5$ and $3x3$ filters. We also added a fully connected layer so that it doesn't go from $3x3x64 = 567$ neurons down to 10 in only one step. To this purpose, we added an intermediate layer of 256 that will then be connected to the 10 neuron output. Under this configuration, we achieved a 89.33% accuracy

1.2 Optimizers

Model	Adam	RMSprop	SGD
Loss			
Accuracy	86.68%	83.51%	42.5%

The conclusion will be that the best Optimizers for our data will be Adam,we also prove other variations of Adam like AdamW,SparseAdam.., but finally we select these 3 to do the comparison.

1.3 Transformations

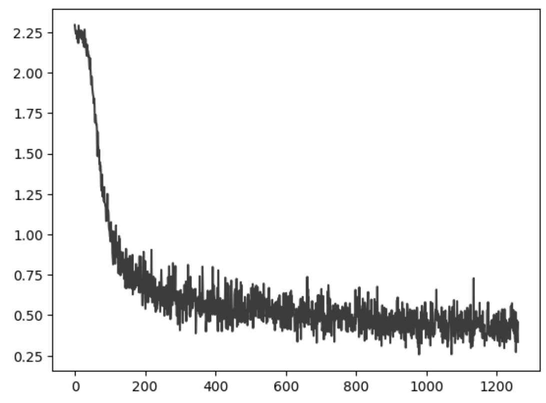
When it comes to performing transformations on the data, we applied 2 methods on top of the previous ones, which were the conversion to tensor and the normalisation. The two methods we tried; cropping and rotation, did not improve the model in any significant way.

Cropping made the accuracy go down a 2%, while the rotation made it basically the same, even though it had a much better loss curve as the previous. We assume this is because we already have plenty of data, and data augmentation is not needed.

1.4 Other Alternatives

Other than this, we applied 2 more changes: schedulers and hard batch mining. The first reduces the learning rate at every epoch, while the second is used to feed the model the batches it performs more poorly in order for it to learn better once it starts to converge and correctly predict most of the images correctly.

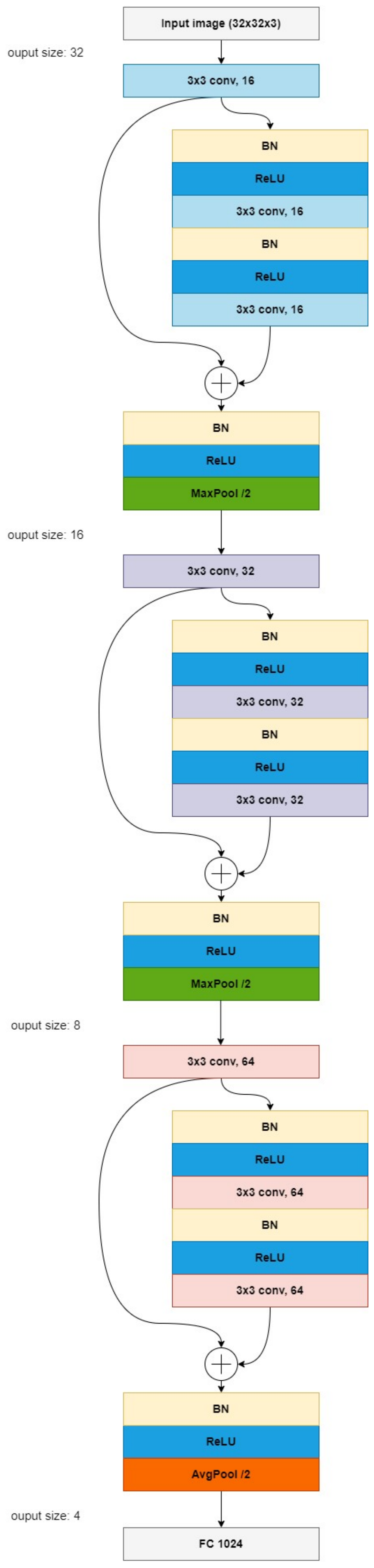
The learning rate scheduler that was applied reduced a learning rate 5% every epoch and didn't produce a considerable increase in performance. This is probably due to the network being quite small and there not being many epochs.



The hard mining did not really have the effect we expected it to. We thought we would see an increase in the loss in the epoch it started (the fourth) but it wasn't the case, probably due to the same reason as before.

The accuracy was maintained.

2. Building a small optimal architecture:



In this exercise we aim to find the best CNN architecture for the SVHN dataset with no more than 150K parameters. There were a plethora of suggestions at the beginning, thus to get an intuition we implemented famous architectures as VGG, Inception and ResNet and analysed their accuracy from scratch. The best results were associated with VGG and ResNet, both had an accuracy around 87%.

Therefore, the main idea of the research was to merge the ideas of both models. Firstly, we implemented a model that had 87290 parameters which consisted of three blocks of two 3x3 convolutions, post activating and then max pooling next connecting it to a FC layer. The initial accuracy was about 86% and after adding skip connections the accuracy increased to 89%. We managed to get better results for example 91% by using the scheduler, Adam optimizer with a learning rate of 0.001 and by *batch harding* that were implemented in the previous exercise. However, this effort could not be considered since the exercise mentions explicitly to not change the hyperparameters.

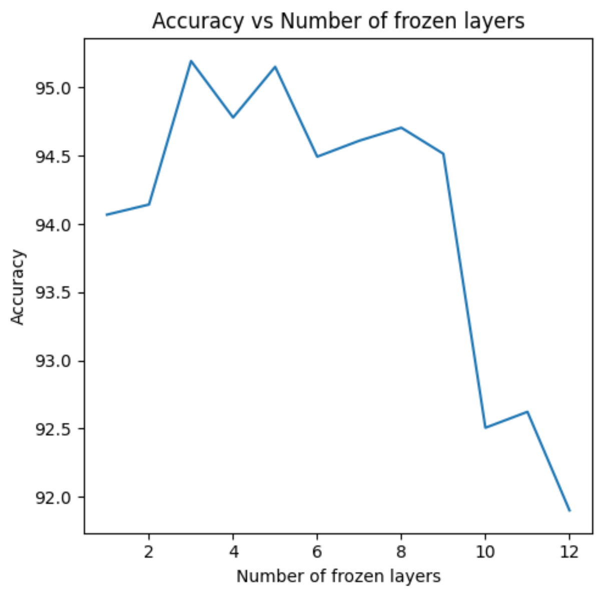
Considering the previous premise, we began to add layers to each block and we ended up with the model that the diagram of the left depicts. We carefully considered the studies at [1] that define the residual blocks. Initially, there was no postactivation after the addition despite the last one because with this configuration the model really benefited from the identity mapping properties such as solving the degradation problem. However, if we considered a Batch normalisation and reLU activation after the addition we get the best results, even though that in [1] states explicitly that this is the opposite of identity mapping since the BN alters the signal layer alters the signal that passes through the shortcut and impedes information propagation. What really happens as [1] states is that the impact of activating after the addition, as in the original ResNet, is not severe enough when it has fewer layers and that is probably the reason why it is working with our model since the degradation problem is not presented. The best results are mainly thanks to the preactivation using BN and ReLU. We also considered to change the convolutions to depthwise convolutions but the performance was worse, as well by implementing fire models. The final accuracy of 5 runs is **93.8%**

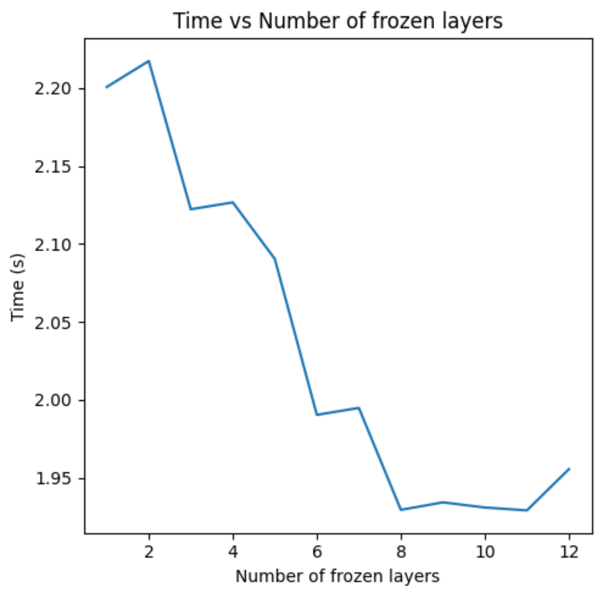
[1]He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Identity Mappings in Deep Residual Networks". arXiv:1603.05027

3.Transfer learning task

The objective of the experiment is to classify a sample of numbers in category (0 or 9), and we will be trying two method, the first pre training the model with a big data set of number from 1 to 8 and then using a small sample of 1´s and 9´s tuned it, or by the way only use this last data sample and analyse the results.

To fine tune the model we have try with different numbers of freeze layers,to compare the accuracy in each case,and also consider the time to process,to finally select the best one in times of accuracy and process time
Accuracy Pre-Trained:





These plots reflect the effects of freezing different numbers of layers (from start to finish) and training only on the unfrozen layers. In our case we decided to freeze each convolution along with it’s corresponding normalisation layer.

The left plot shows how the accuracy of the fine-tunned model decreases along with the number of trainable layers. This is most likely due the model training less parameters than it needs in order to learn the new distribution. We can also see how at first the model underperforms with respect to its potential. We interpret this to be because of there being too many parameters to train on such few samples of 0's and 9's and therefore overfitting occuring. Other than this we have another alternative explanation which is that the features extracted at each layer get increasingly more complex and specific, and maybe the initial feature extractions are already very well tunned to extract the general features of the images.

The right plot shows how compute-time is affected by the decrease in parameters to be optimised. It comes as no surprise that as the number of parameters increases (more values to perform optimisation on) the time it takes for the training does too.

Accuracy not Pre-Trained:69,43
Training a neural network with a small dataset that only contains the numbers 0 and 9 can lead to overfitting. Adding a larger dataset that includes numbers 1 to 8 can help mitigate this issue.

By providing more examples of data, the neural network can learn more general patterns and improve its ability to generalise.

When the neural network is trained with a more diverse dataset that includes numbers 1 to 8, it learns to recognize common features among different digits and can make more accurate generalisations about which features are relevant for classifying the numbers.

Furthermore, by introducing numbers 1 to 8 in the training set, the neural network gains a better understanding of the context in which the numbers 0 and 9 appear. This can help prevent the network from overfitting to specific features of the numbers 0 and 9 that are not relevant for their classification.