Abstract

This project examined the usage of AlexNet[1], a network built for classification of objects using the ImageNet database, in the training on the '1P' Dataset from the IllustrisTNG simulation of the CAMELS Multifield Dataset (CMD)[2]—a task that is regression based. It was found that when working only with magnetic field maps, most of the architecture of AlexNet could be preserved and lead to converging loss. Specifically, training, validation, and test errors were reported as 0.1007, 0.0981, and 0.0981 respectively. It was found that this same network was more prone to overfitting when all fields are considered simultaneously. The best training, validation and test errors that could be found were, 0.0991, 0.1394, and 0.1115. This report dicusses if there is merit in considering more classification based architecture for regression based tasks, as well as what other examples of such architecture may look like.

1 Introduction

The Cosmology and Astrophysics with MachinE Learning Simulations (CAMELS) Multifield Dataset, CMD[2], is a publicly available collection of hundreds of thousands 2D maps and 3D grids containing different properties of the gas, dark matter, and stars from more than 3,000 different universes. In this work, a section of this data: the '1P' dataset from the IllustrisTNG simulation, will be used to train a neural network to recognize astrophysical parameters. The '1P' dataset contained 2D Maps for various astrophysical fields, shown in table 1.

Field	**Prefix**
Gas density	Mgas
Gas velocity	Vgas
Gas temperature	T
Gas metallicity	Z
Neutral hydrogen density	HI
Electron number density	ne
Magnetic fields	B
Magnesium over Iron	MgFe
Dark matter density	Mcdm
Dark matter velocity	Vcdm
Stellar mass density	Mstar
Total matter density	Mtot

Table 1: '1P Dataset contains equal 2D Maps for multiple astrophysical fields. Each field was referred in the project by a given Prefix.[2]

Each 2D Map is controlled by six parameters, which are varied one-by-one, for a total of 66 different sets of parameters. These parameters consist of two important cosmological parameters, σ_8 and Ω_m (fraction of matter in the universe and smoothness of the distribution of matter in the universe respectively.), and four parameters $A_{SN1}, A_{SN2}, A_{GN1}, A_{GN2}$(supernova and black-hole feedback) that might need to be considered to determine the interesting cosmological parameters. There are 15 maps present for each parameter combination for every astrophysical field. [2]

Much of the literature present in image-based CNN networks are used for classification. This report will examine how one such existing architecture, AlexNet[1], generalises to a regression based task such as this one.

This project was split into two parts. First, a network was trained only on the maps for magnetic fields, with testing and validation error being examined for the same data. Then, all astrophysical fields were considered together and another network was trained on this data.

2 System Requirement

The following networks were ran on a high-RAM setup in Google Colab Pro. This allowed the project 51 GB of RAM. Although Colab Pro allows for the use of GPU accelerators, these were not used.

3 Task 1: Network Trained on Magnetic Fields

3.1 Data Preparation

Out of the fifteen maps that each parameter value corresponded to, the first nine, were made part of the training set, the next three were made part of the validation set, and the last three were made part of the test set. This allowed for all sets to have a distribution of data that would avoid bias in the model.

In an effort to decrease loss and create more data, data augmentation was performed on the training data. Three augmented datasets were formed using keras.reverse(axis =). This lead to four times the data that was initially present.

3.2 Architecture

This project used a modified version of AlexNet(2012)[1] that was changed to account for the amount of data and prevent overfitting to the train data. Efforts were made to conserve the overall-process of slowly creating a layer that is concentrated and has many filters. AlexNet is shown in figure 1, while the network architecture for this task is shown in figure 2.

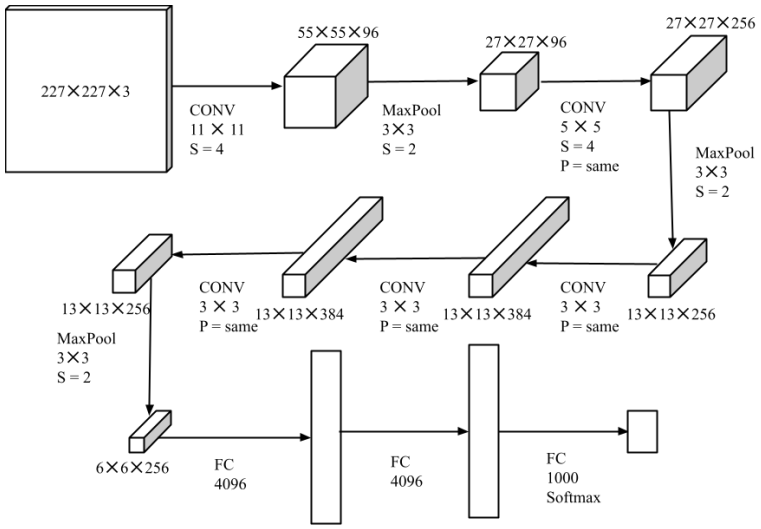

Figure 1: AlexNet[1]

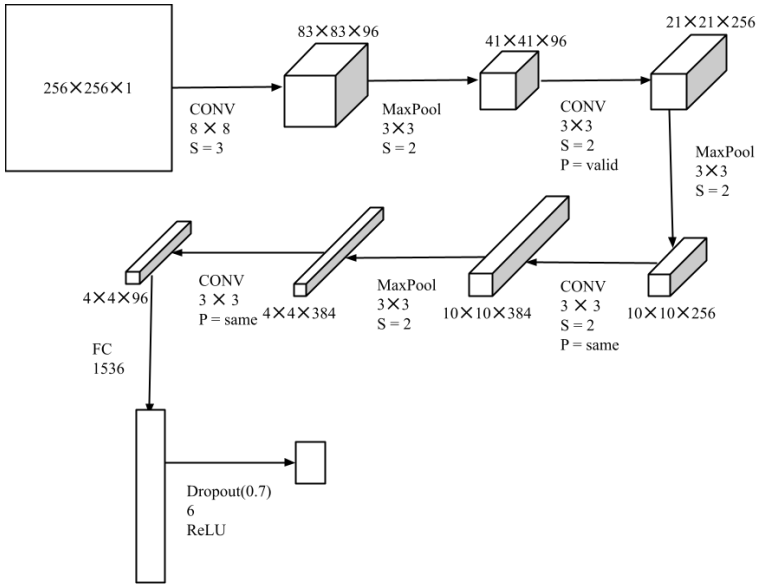

Figure 2: Modified architecture for Task 1. This was found through trial and error, taking AlexNet as the starting architecture.

It was obvious from the very first implementation that AlexNet was too robust of a network to train the magnetic fields data, which was nowhere close to the size the ImageNet database, for which AlexNet was initially created. As such, the number of convolutions were reduced while still preserving the increasingly narrowing quality of AlexNet. A dropout of 0.7 was also included into the model—this is further discussed in section 3.4.

3.3 Optimizer, Loss, and Metric

The optimizer used for this network was SGD, with a learning rate 10^{-2} and a momentum of 0.9. This was found to work the best after multiple iterations. As this model was a regression based, the loss was chosen to be mean squared error. This seemed to provide the most optimal performance out of all ready regression-based losses provided by keras. Finally, due to the regression based loss of the task, 'accuracy' could not used as a metric. As such, the loss described earlier was used as the metric. It would be known that the model had been well trained if the training, valid, and test loss seemed to converge around a loss value.

3.4 Results

This network converged at a validation error of 0.0981, with a final training error of 0.1007. The higher training error was caused due to dropout which at 0.7, was higher than normally used. This was concluded as the model was ran without dropout, at which point training and validation losses converged to the same value. Using this high dropout value did not seem to disturb the validation error, and even sped up the training, with each epoch taking <30s with a batchsize of 32. The test accuracy was reported as: 0.0981. Figure 3 shows the loss of this network over 10 epochs. Interestingly, The network was tested without the augmented data and there was no change observed in the final loss.

4 Task 2: Network trained on all maps

4.1 Data Preparation

The train, validation, and test splits were kept the same as task 1. The splitting as well as the data augmentation were run through all maps iteratively to form a dataset with 28512 values.

4.2 Architecture

Although this section had many more data points, it was found that the network used in task 1 was severely overfitting to the training data, with the validation error being 10 times as large as the training error. As such, the network was made smaller. Figure 4 shows the architecture of the network. Another element

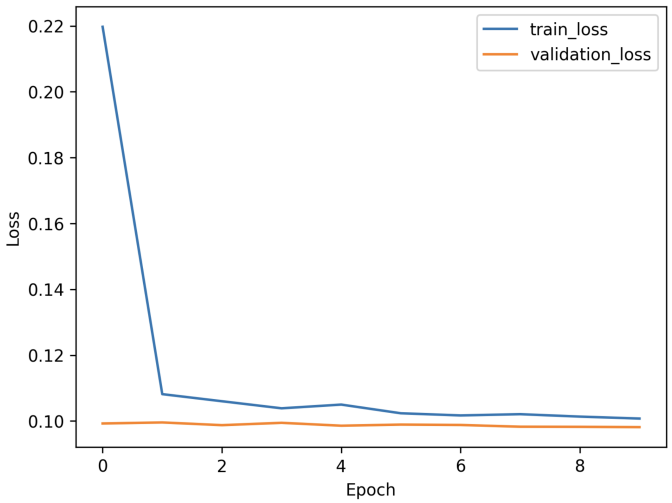

Figure 3: Train and Validation loss during training of Task 1. Both values converge to the same loss, train loss being higher throughout due to dropout.

that needed to be added to network, was batch normalisation. Without batch normalisation, the gradient would explode immediately, causing the loss to go to 'NaN'.

4.3 Optimizer, Loss, and Metric

The loss and metric were kept unchanged from the first task. The optimizer was changed to 'Adam' as for this task it seemed to work better. This could be because of the model's need to update the learning rate due to vast amount of data being handled.

4.4 Results

This network converged at a training error of 0.0991, with a final validation error of 0.1394. This was a worse performance than the network in Task 1, which was unexpected. Despite multiple attempts to change the network, there was a good chance it had been overfit to the training data. To stop the model from fitting any more than it needed to, training was stopped at 60 epochs, as validation error seemed to be noisy and mostly increasing past this point. There was no dropout included in this network as it seemed to worsen performance. Opposite to the first task, augmenting data made a huge difference to the validation error—bringing down validation loss by almost 900 percent. Compared to the first task, this network was much slower, taking up to six minutes per epoch. The test loss was reported as: 0.1115. Figure 5 shows the loss of this network over 60 epochs.

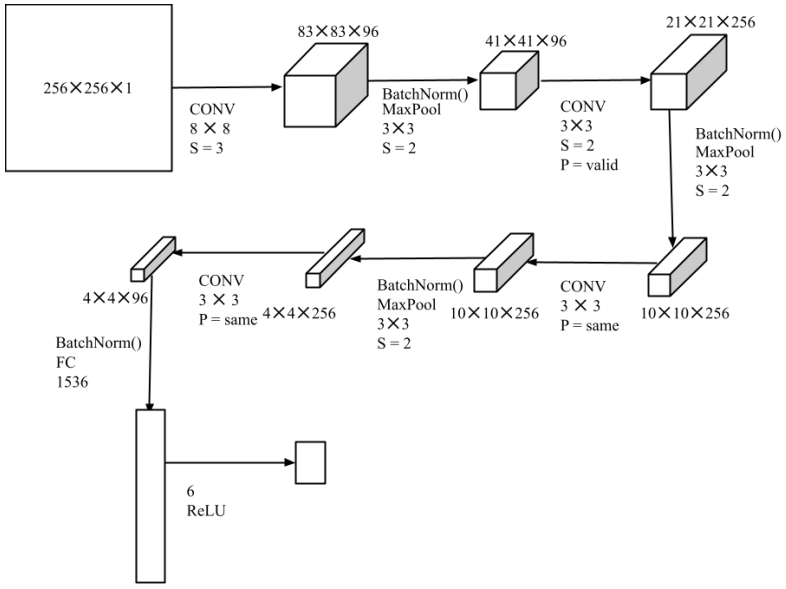

Figure 4: Modified architecture for Task 2. This was found through trial and error, taking AlexNet and the network found in task 1 as the starting architecture.

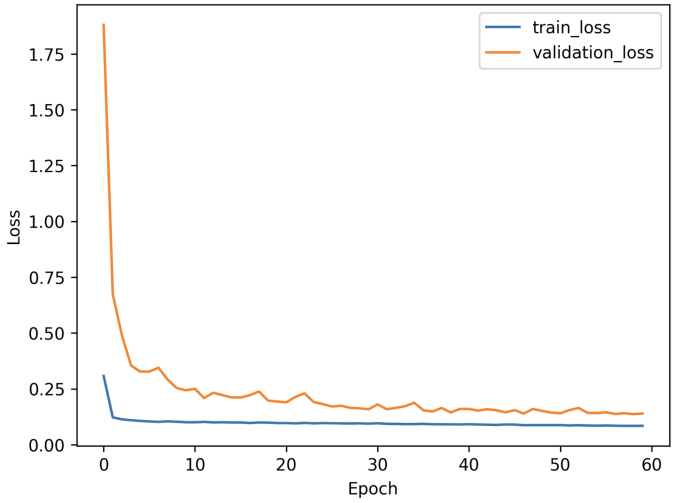

Figure 5: Train and Validation loss during training of Task 2. Both values converge to the same loss, validation loss being higher throughout due to slight overfitting.

5 Conclusion

After using AlexNet[1] to find parameter values in the '1P' dataset, it can be concluded that variations of AlexNet do well when learning the parameters provided from the maps. The network seemed to learn better when it only had learn and predict from maps of one astrophysical field, with the magnetic field network having training, validation, and test errors of 0.1007, 0.0981, and 0.0981 respectively. Compared to this network that learned and had to predict from all maps, these values were slightly higher, with training, validation and test errors being, 0.0991, 0.1394, and 0.1115. There are perhaps better network architectures to handle this dataset that would not provide errors like overfitting and an exploding gradient. As an extension, it may be useful to check ensemble models or other classification architectures like Google LeNet, which are known to prevent overfitting due to their sparsely connected structures.

References

[1] Krizhevsky Alex, et al.(2012)ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems.*25. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[2] Villaescusa-Navarro, et al. 2021, arXiv e-prints, arXiv:2109.1091. https://arxiv.org/abs/2109.1091