Deep Learning Assignment #1

Michał Janiszewski
MEEC — 65637
NOVA School of Science and Technology
Caparica
m.janiszewski@campus.fct.unl.pt

Artur Stopa
MEI — 65043
NOVA School of Science and Technology
Caparica
a.stopa@campus.fct.unl.pt

Abstract

Before solving tasks from assignment #1 class balance was checked and there is no significant class imbalance. Objective of the task is to implement and evaluate DNN for purpose of image classification and segmamtation. Dataset consists of pokemon figures layered on top of real life photos takes around Nova FCT campus.

1 Multi-Layer Perceptron

1.1 MLP Introduction

As expected simple multi-layer perceptron performance on our image classification task is not reliable reaching plateau of accuracy at about 30% which could be linked to models inability to understand spatial relationships between image pixels without understating more complex patterns in the provided data as well as having trouble with larger amounts of parameters due to providing it higher dimensional objects flattened into series.

Figure 1: Multi-Layer Perceptron architecture

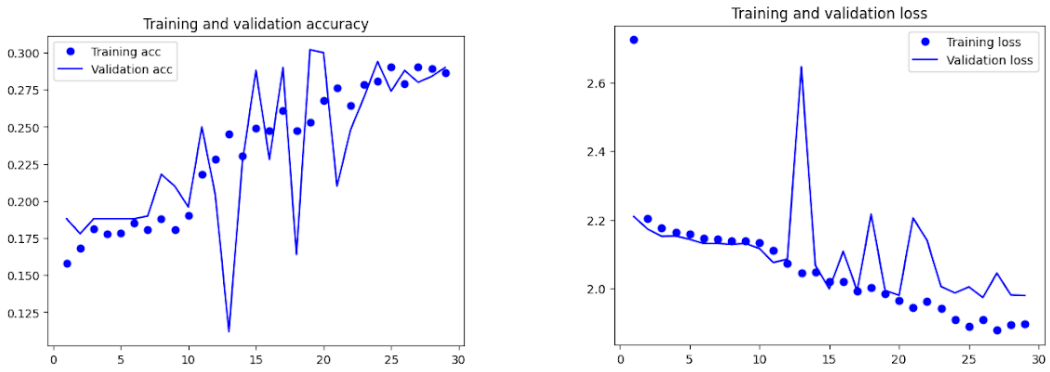
1.2 MLP Process

After initial exploration of effects which tuning model parameters like learning rate ,number of layers/neurons ,batch size and optimizers we came to the conclusion that MLP was prone to underfitting the provided data even after increasing model complexity and introducing multiple regularisation techniques like batch normalisation,L2 kernel regularisation and dropout layers we couldn't improve its performance as accuracy obtained by the model appeared to be random would often get stuck on local minimas across its gradient or go spanning its accuracy on test set in whole range from 5-40%.

1.3 MLP Callbacks

Due to the instability of the model we introduced dynamic callbacks which would react to its performance in real time. EarlyStopping- In order to preserve weights of model for which validation accuracy was highest ReduceLRonPlateau- In order to adjust learning rate actively when the model got stuck on local minima and therefore could be navigated out of it. Callbacks stabilised our model but did not provide much improvement however they were later used in other architectures like CNN and proved to boost performance significantly.

Preprint. Under review.



Multi-Layer Perceptron accuracy

Multi-Layer Perceptron loss

Figure 2: Multi-Layer Perceptron graphs

1.4 MLP and CNN optimizer and loss function

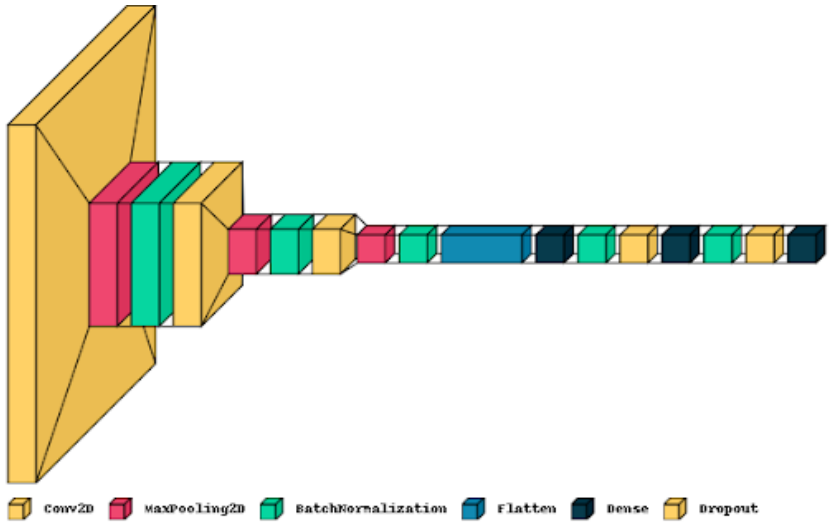
Chosen loss function for both MLP and CNN was categorical cross-entropy due to the fact that it was multiclass classification . After exploring some other optimizers like SGD and Adagrad we came to the conclusion that they did not improve performance due to the fact they got stuck on local minima quite often which led us to choose Adam in the end. When it comes to activation function it should be no surprise that ReLU was used to introduce non-linearity and prevent vanishing gradients.

2 Convolutional Neural Network

2.1 CNN Introduction

CNN possesses higher capacity and thus larger computational power and learning time due to overall complexity, multiple convolution and pooling layers for feature extraction. Thanks to them convolutional layers can understand spatial relationships between pixels and help the model understand much more complex patterns in the data.

Figure 3: Convolutional Neural Network architecture

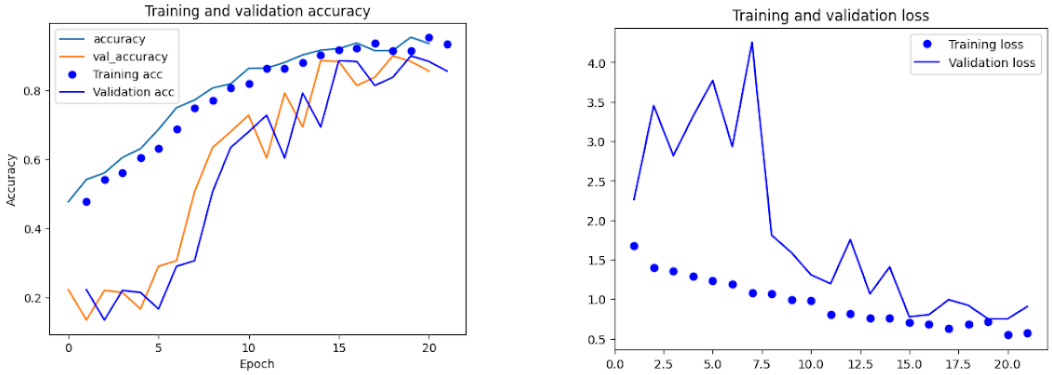


2.2 CNN exploration

After Initial exploration done with a very large model with multiple dense and convolution layers each one having around 128 or 256 neurons on higher learning rate without much regularisation we obtained a model which could easily overfit the data which could be seen on rapidly growing training accuracy and enormous fluctuations in validation accuracy which ranged from 9% - 85% thus showing that convergence was difficult to achieve with such a model. This led us to decrease model complexity reducing number of layers and neurons to 32 or 64 , adding multiple Batch Normalisation and L2 kernel regularisation across both convolution and fully connected layers of our model with addition of Dropout layer in FC part.

2.3 CNN with regularization

On the provided graphs we can see quite a lot of fluctuations of validation accuracy which suggests that gradient of our model is very uneven and has multiple extrema minima across its span which combined with initial learning rate close to lr=10-2 led us to get stuck on some of them but after further improvements we managed to obtain model that performs well on given dataset giving accuracy of around acc 80-90% on both validation set. Test accuracy was 83%.



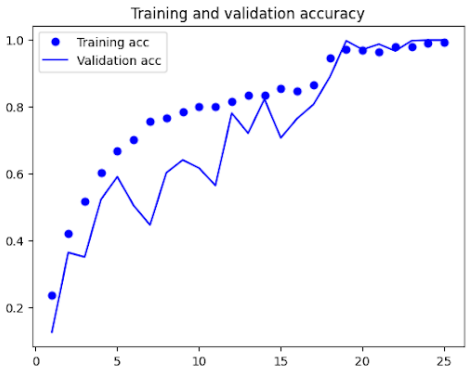
Convolutional Neural Network accuracy

Convolutional Neural Network loss

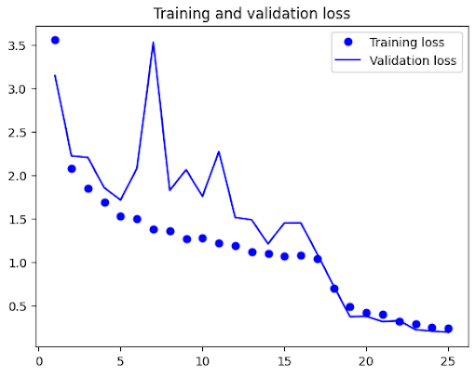
Figure 4: Convoltional Neural Network graphs

2.4 CNN with regularization and callbacks

After introducing ReduceLRonPlateau and EarlyStopping callback functions and optimising their parameters for our CNN our model learned to navigate around local minimas due to dynamic learning rate reduction around those areas and managed to reduce validation loss extremely close to global minima giving us almost perfect algorithm for the given dataset. It is important to keep in mind that while it works wonderfully on provided test data it is most likely due to the fact that images of pokemons in both training and test data are extremely similar to one another. On more diverse datasets the callback function would most likely have to be used much more carefully to prevent it from overfitting given training data. Log of each epoch provides us a more insightful look into models operation - after getting stuck on plateau while increasing validation accuracy it decrease learning rate by multiplying it by our arbitrary factor=0.2 after which significant improvement can be seen. Test accuracy was 99,5%.



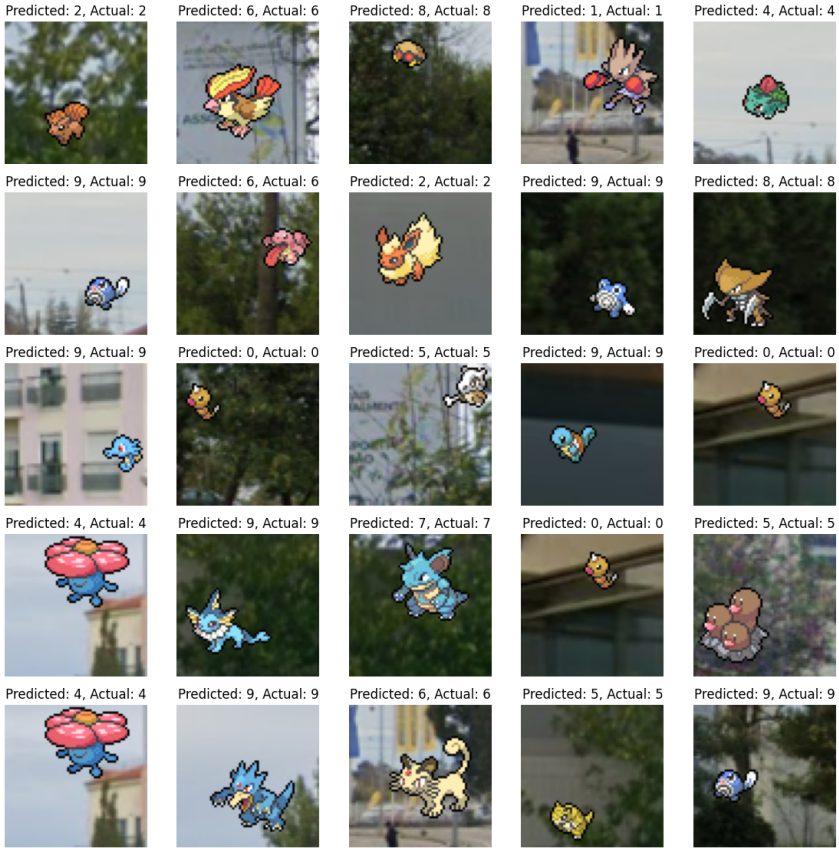
Convolutional Neural Network accuracy



Convolutional Neural Network loss

Figure 5: Convolutional Neural Network graphs

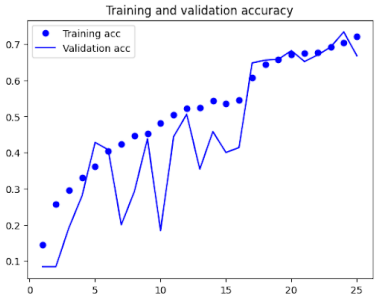
Figure 6: CNN predictions visualisation



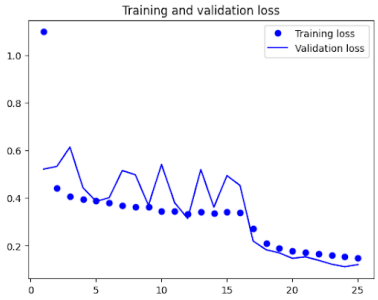
3 Multilabel classification

After experimenting with different architectures and seeing no valid improvements we decided to use our final version of CNN with some twists for this task. As a loss function binary cross-entropy was used in order to penalise each pokemon type error independently of what happens to the others. Output layer activation function was set to sigmoid in order to give activation in range [0,1]. More Dropout layers were introduced and batch size was decreased to 16 to increase gradient update and to allow model to generalise better on new data. Without callbacks the model performed at around 50-60% accuracy with the callbacks it reached 70% which is a satisfying result however it possibly

could be improved with much more powerful feature extraction process which will be done in the next part of the assignment.



Multilabel Convolutional Neural Network accuracy



Multilabel Convolutional Neural Network loss

Figure 7: Multilabel CNN graphs

4 Transfer learning

Transfer learning utilizes a network pre-trained on a big dataset as a feature extractor. To do so it's necessary to remove top layers of this network, as they are responsible for classification of data points, while the lower layers ones extract features from raw data. Lowest layers extract the most general features with higher ones extracting more abstract and specific ones progressively. Usually when using state-of-the-art networks like ResNet or MobileNet it's enough to remove only the top layers, even when datasets differ vastly (like in this case - real photos of imagenet and pixel-art pokemon of our problem). Sometimes it might be necessary to take features from deeper layers, but in this case it was not. FCT pokemon dataset has shape of 64x64x3, while keras.applicaitions models have pretrained weights for 224x224x3 shape, thus the images had to be resized. MobileNet model was used as a feature extractor because it's a small model designed to run on mobile devices and FCT pokemon problem is a simple one and it should work well in this situation.

4.1 Single label multiclass classification

4.1.1 Classificator training

For single label multiclass classification categorical crossentropy loss funciton was used with 10 output neurons that have softmax activation function. Every neuron represents it's belonging to a particular class, but thanks to softmax activation function only one of the output neurons will be activated. MobileNet is fully frozen.

Early stopping and reducing learning rate on plateau callbacks were used. Validation split was =0.125. Test accuracy of accuracy 0.9400 was achieved

4.1.2 Fine-tuning

Last 10 layers (6 of which had trainable parameters) of MobileNet feature extractor were unfrozen for fine tuning and learning rate was reduced by a factor of 10. Loss function was RMSprop, because it reduces learning rate after going through several batches. Rest of the hyperparameters was unchanged. The same early stopping reducing learning rate on plateau was used. Test accuracy of accuracy 0.9960 was achieved.

4.2 Multilabel multiclass classification

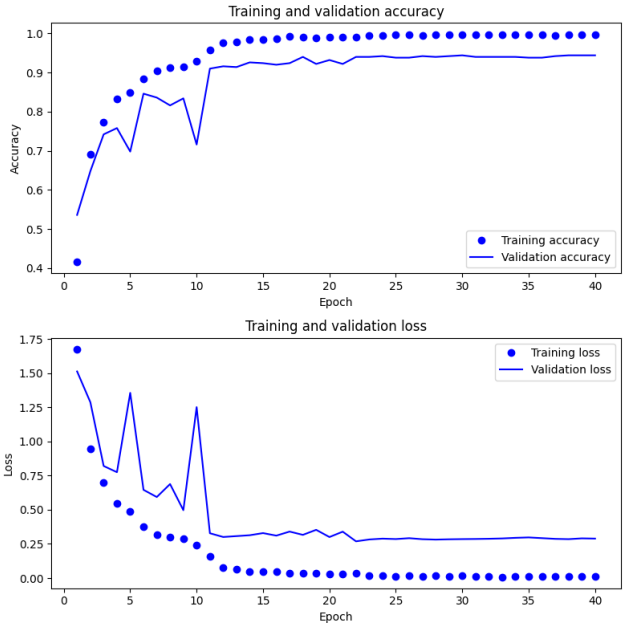
4.2.1 Classificator training

For multilabel multiclass classification binary crossentropy loss funciton was used with 10 output neurons that have sigmoid activation function. Every neuron represents it's belonging to a particular class or not, which thanks to sigmoid activation is independent of it's belonging to any class. Binary

Figure 8: Single label multiclass model summary

```
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)        3228864
nal)

global_average_pooling2d (G  (None, 1024)              0
lobalAveragePooling2D)

dense (Dense)                (None, 256)               262400

batch_normalization (BatchN  (None, 256)               1024
ormalization)

dropout (Dropout)            (None, 256)               0

dense_1 (Dense)              (None, 128)               32896

dropout_1 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 128)               16512

dropout_2 (Dropout)          (None, 128)               0

dense_3 (Dense)              (None, 64)                8256

dense_4 (Dense)              (None, 10)                650

=================================================================
Total params: 3,550,602
Trainable params: 321,226
Non-trainable params: 3,229,376
_________________________________________________________________
```

Figure 9: Single label multiclass model accuracy and loss



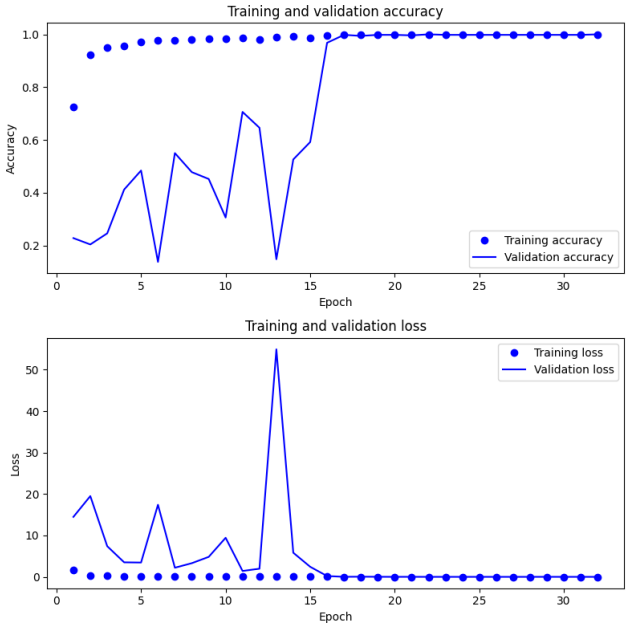
crossentropy optimizes prediction of belonging to class C or belonging to class not C for all ten possible classes. MobileNet is fully frozen.

Early stopping and reducing learning rate on plateau callbacks were used. Validation split was =0.125. On test set accuracy of 0.8140 and recall of 0.9438 was achieved. We know that dataset isn't significantly imbalanced but recall was checked anyway.

Figure 10: Single label multiclass fine-tuned model summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)       3228864
nal)

global_average_pooling2d (G  (None, 1024)             0
lobalAveragePooling2D)

dense (Dense)                (None, 256)              262400

batch_normalization (BatchN  (None, 256)              1024
ormalization)

dropout (Dropout)            (None, 256)              0

dense_1 (Dense)              (None, 128)              32896

dropout_1 (Dropout)          (None, 128)              0

dense_2 (Dense)              (None, 128)              16512

dropout_2 (Dropout)          (None, 128)              0

dense_3 (Dense)              (None, 64)               8256

dense_4 (Dense)              (None, 10)               650

=================================================================
Total params: 3,550,602
Trainable params: 1,909,450
Non-trainable params: 1,641,152
_________________________________________________________________
```

Figure 11: Single label multiclass fine-tuned model accuracy and loss



4.2.2 Fine-tuning

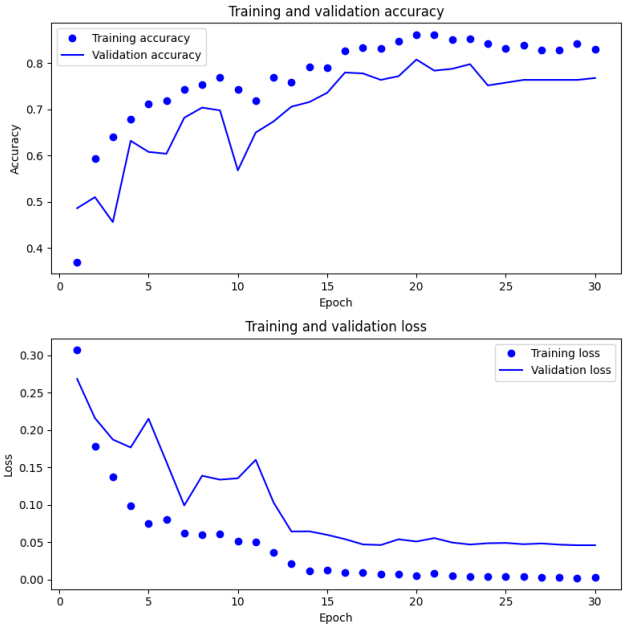
Last 10 layers (6 of which had trainable parameters) of MobileNet feature extractor were unfrozen for fine tuning and learning rate was reduced by a factor of 10. Loss function was RMSprop, because it reduces learning rate after going through several batches. Rest of the hyperparameters was unchanged. On test set accuracy of 0.9480 and recall of 0.9986 was achieved. The same early stopping reducing learning rate on plateau was used.

za

Figure 12: Multilabel multiclass model summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)       3228864
nal)

global_average_pooling2d_2   (None, 1024)            0
(GlobalAveragePooling2D)

dense_10 (Dense)             (None, 256)             262400

batch_normalization_2 (Batc  (None, 256)             1024
hNormalization)

dropout_6 (Dropout)          (None, 256)             0

dense_11 (Dense)             (None, 128)             32896

dropout_7 (Dropout)          (None, 128)             0

dense_12 (Dense)             (None, 128)             16512

dropout_8 (Dropout)          (None, 128)             0

dense_13 (Dense)             (None, 64)              8256

dense_14 (Dense)             (None, 10)              650

=================================================================
Total params: 3,550,602
Trainable params: 321,226
Non-trainable params: 3,229,376
_________________________________________________________________
```

Figure 13: Multilabel multiclass model accuracy and loss



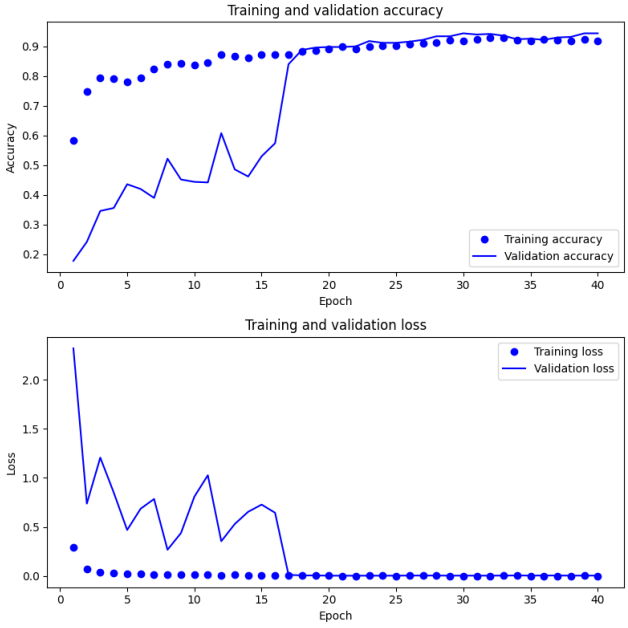
4.3 Observations

Data augmentation significantly lowered model's accuracy and learning speed so in the end it wasn't used. Above results show that it was unnecessary anyway. Multilayer perceptron on top of featue extractor learns so fast that it doesn't make sense to attempt shrinking the model. Dataset we have in this problem is very specific and different to real images from imagenet, thus it's not surptising that training our own feature extractor works slightly better than using only feature extraction for a single label classification. Fine-tuning obviously outperforms both. For a more complex problem

Figure 14: Multilabel multiclass fine-tuned model summary

```
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)       3228864
nal)

global_average_pooling2d (G  (None, 1024)             0
lobalAveragePooling2D)

dense (Dense)                (None, 256)              262400

batch_normalization (BatchN  (None, 256)              1024
ormalization)

dropout (Dropout)            (None, 256)              0

dense_1 (Dense)              (None, 128)              32896

dropout_1 (Dropout)          (None, 128)              0

dense_2 (Dense)              (None, 128)              16512

dropout_2 (Dropout)          (None, 128)              0

dense_3 (Dense)              (None, 64)               8256

dense_4 (Dense)              (None, 10)               650

=================================================================
Total params: 3,550,602
Trainable params: 1,909,450
Non-trainable params: 1,641,152
_________________________________________________________________
```

Figure 15: Multilabel multiclass model accuracy and loss



like multilabel classification our dataset is too small for our own feature extractor to outperform a network like MobileNet even without fine-tuning, but with big enough dataset it should be possible.

5 Class Activation Mapping

Activations of each convolutional layer showcased to get deeper into algorithms way of understanding the complex patterns present in the data. We can see what second iteration would most likely be sufficient as Max pooling and filters decrease amount of detail in spatial map in the 3rd one, however

the provided CNN architecture got 99.6% accuracy on test data. Moreover from preformed tests we came to the conclusion that 3rd layer helped with invariance of translation ability of the model.

Figure 16: Class activation mapping