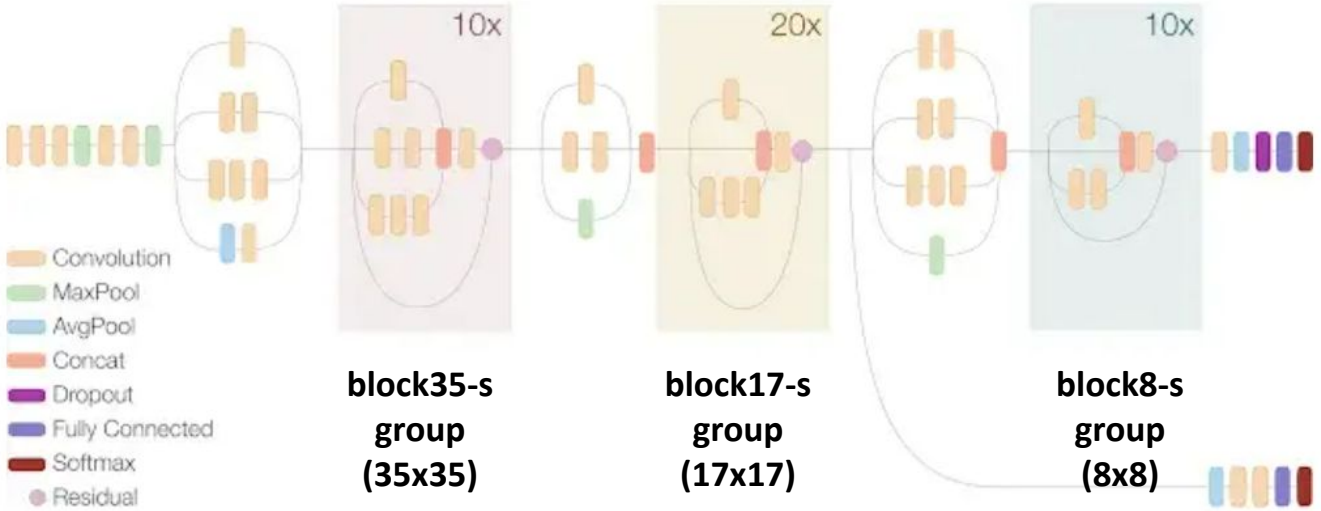
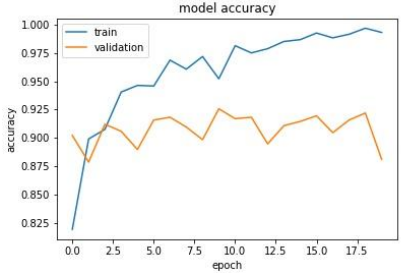
TASK 0: Fine tuning existing model

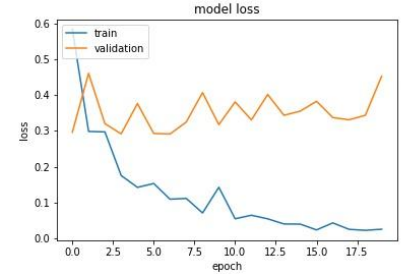
InceptionResnetV2: Resnet with residual connections



TASK 0: Fine tuning existing model

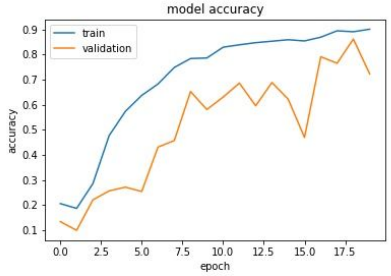
All frozen until last one

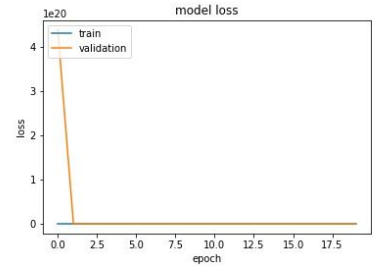




Val. acc.: 0.92

No one frozen





Val. acc.: 0.88

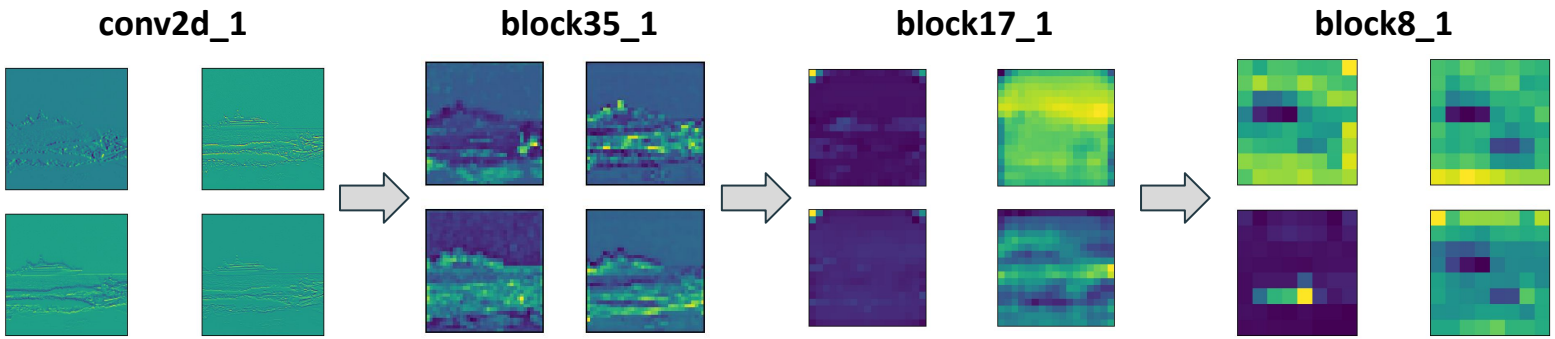
Overfitting is indicated by the fact that accuracy and loss are much higher during training than during validation. This is particularly evident when we freeze all layers except the last one. We observe low accuracy and high loss during the first epoch when we do not freeze any layers, as we are retraining the entire neural network.

In the following slides, we will explore methods to avoid overfitting and improve accuracy and loss on validation datasets.

TASK 0: Fine tuning existing model

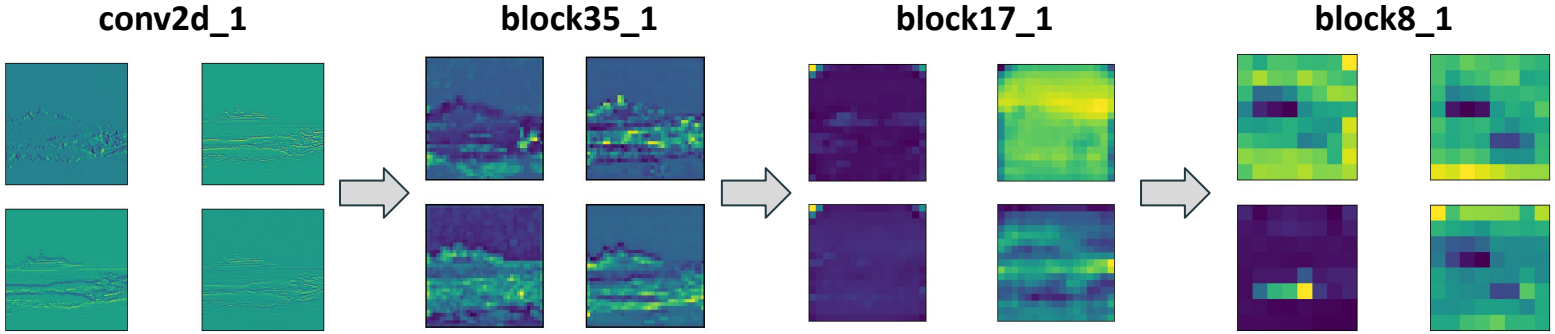
Feature map extraction through different layers

In the visualization of feature maps, the third dimension corresponds to the number of feature maps. To visualize a single feature map, you can simply select one and remove the third dimension using the numpy's squeeze() function, which eliminates single-dimensional entries from the shape of an array.



TASK 0: Fine tuning existing model

Feature map extraction through different layers



It can be observed that the feature maps closest to the input image are more similar to the original image and contain fewer representational features. However, in deeper layers of the network, more abstract features and less similarity to the original image can be observed.

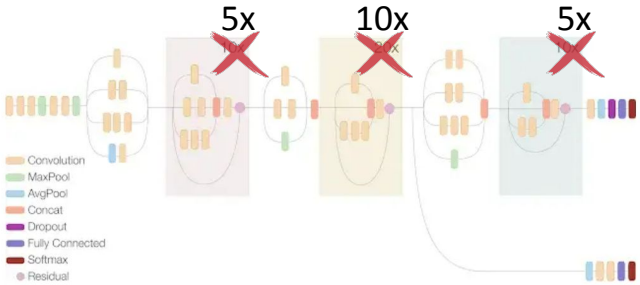
TASK 1: Set a new model

As our dataset is not very large, we do not require networks as large as InceptionResnetV2. Therefore, we will reduce the number of layers using two methods:

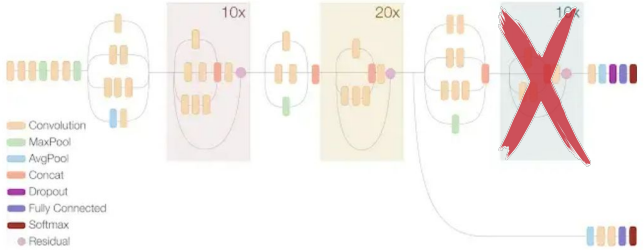
1. By removing some blocks within a block group while maintaining the image size flow in the network.
2. By removing block groups from the end of the network.

We will train all the models using both of these methods, gradually removing more blocks per training and observing their performance.

Method 1:



Method 2:



TASK 1: Set a new model

Removing blocks method 1:

Removed blocks	Blocks left	Partition of remove (per group)	Parameters	Validation Accuracy
0	40		54 M	87,6
4	36	1|2|1	50 M	89,7
8	32	2|4|2	45 M	89,5
16	24	4|8|4	36 M	89,1
32	8	8|16|8	19 M	88,9
36	4	9|18|9	14 M	89,1

It can be observed that removing a significant number of blocks from the network does not result in a loss of performance. This indicates that our dataset does not require such a complex network.

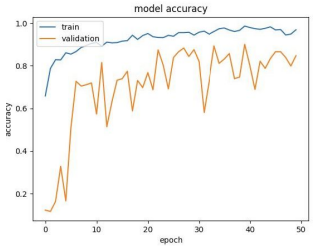
TASK 1: Set a new model

Removing blocks method 2:

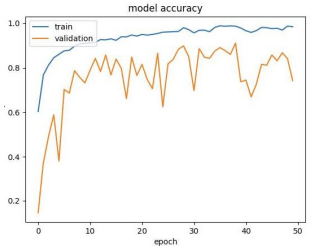
Removed group	Parameters	Validation Accuracy
Block8-s	32785384	0.92
Block17-s	5478408	0.91
Block35-s	**567896**	**0.90**

This approach resulted in fewer parameters and better results. With only 500k parameters, we achieved 90% accuracy on the validation set. We will use this new network for the next tasks.

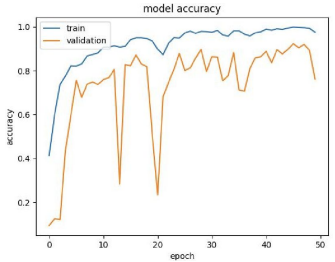
No block8-s:



No block8-s and block17-s:



No block8-s, block17-s and block35-s:

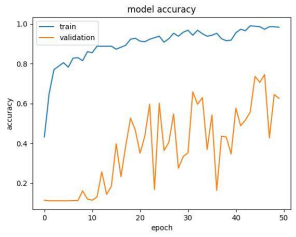


TASK 2: Apply small dataset

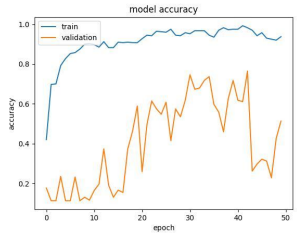
The obtained new short network will be trained with a small dataset of only 400 images for training. Several partitions will be trained:

Dataset	Train Accuracy	Validation Accuracy
Large dataset	0.99	0.9
MIT_Small_train1	0.98	0.745
MIT_Small_train2	0.99	0.76
MIT_Small_train3	0.98	0.76
MIT_Small_train4	0.98	0.779

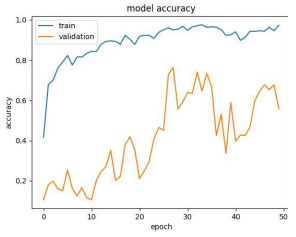
Small_train1



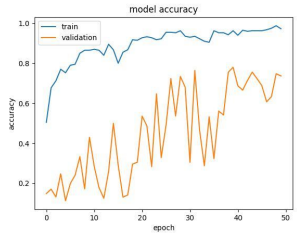
Small_train2



Small_train3



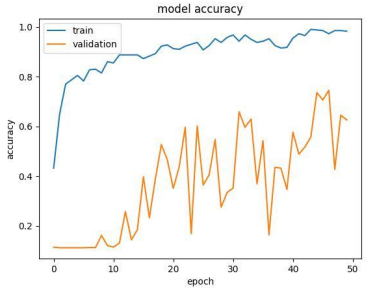
Small_train4



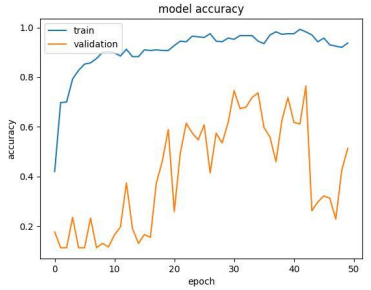
TASK 2: Apply small dataset

It can be observed that using a smaller training dataset results in the network overfitting more, as the difference between training accuracy and validation accuracy is larger. To combat this, we will implement data augmentation techniques. Our next tasks will be performed using this first small dataset.

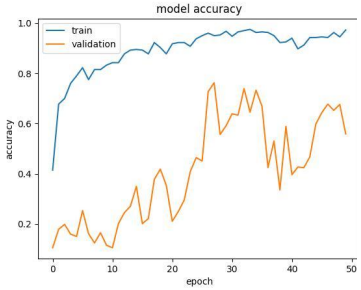
Small_train1



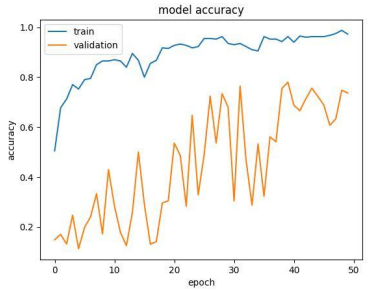
Small_train2



Small_train3



Small_train4



TASK 3: Data augmentation

As a regularization method, some data augmentation techniques will be tried. These transformations of the images will be applied randomly to the train images, but never to the validation images.
The following techniques will be used:

Original:

Horizontal Shift (H_S):

Vertical Shift (V_S):

Horizontal Flip (H_F):

Vertical Flip (V_F):

Rotation (R):

Brightness (B):

Zoom (Z):

Shear (S):

TASK 3: Data augmentation

Single:

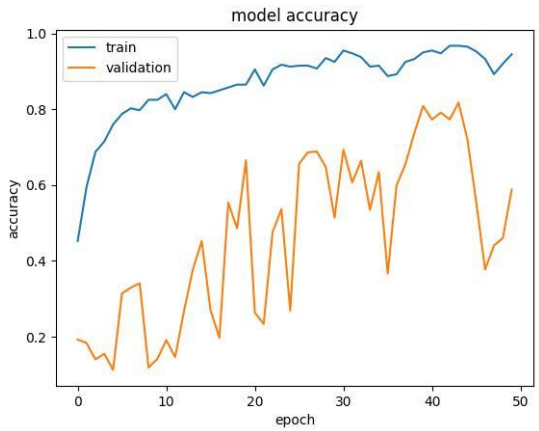
Data augmentation techniques	Train accuracy	Validation accuracy
None	0.98	0.745
W_S	0.97	0.749
H_S	0.975	0.745
H_F	0.98	0.762
V_F	0.98	0.701
R	0.96	0.738
B	0.97	0.785
Z	0.98	0.745
S	1	0.821

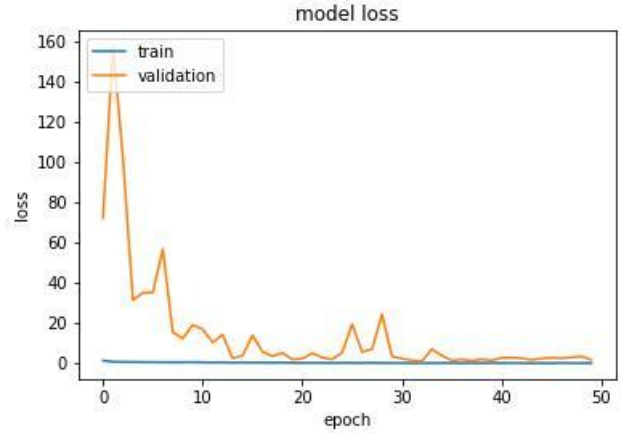
Combinations:

Data augmentation techniques	Train accuracy	Validation accuracy
None	0.98	0.745
W_S + H_F	0.97	0.780
W_S + B	0.97	0.768
W_S + S	0.98	0.784
H_F + B	0.98	0.763
H_F + S	0.97	0.778
B + S	0.97	0.818
W_S + H_F + B + S	0.95	0.799

TASK 3: Data augmentation

Best result using only Shear transformation (82%):





The combinations of data augmentation techniques did not help to avoid overfitting. The best data augmentation technique was using only Shear transformation, which improved our accuracy from 75% to 82%.

However, it can be observed that the validation loss curve is not smooth and spikes in the initial epochs. We will try to address this issue in the next task.

TASK 4: Improve learning curve:

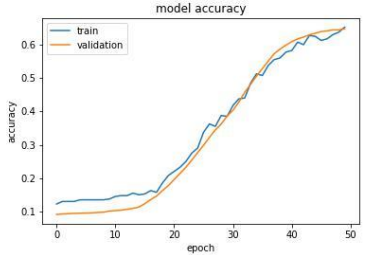
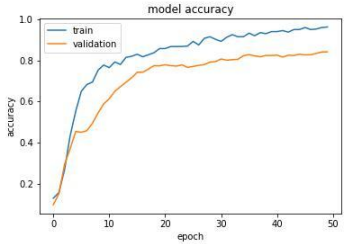
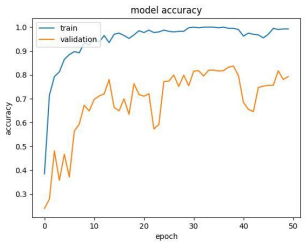
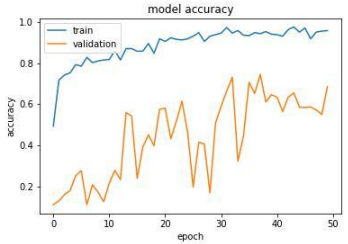
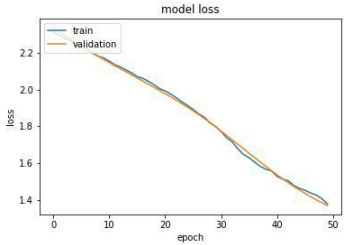
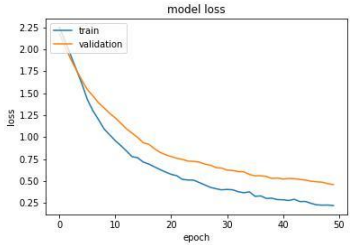
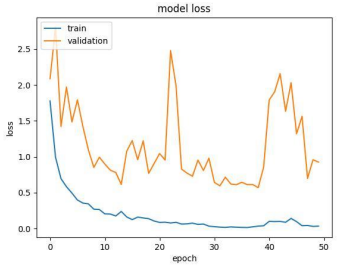
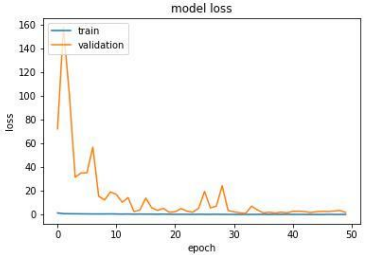
In order to improve our learning curves, we will try three different methods:

1. Experiment with different learning rates in order to prevent the validation loss from exploding in the initial epochs and to achieve smoother curves.

2. Replace Batch Normalization layers with Layer Normalization, as our model is using Batch Normalization layers and our batch size is not too large, the calculated mean and standard deviation may not be representative enough.

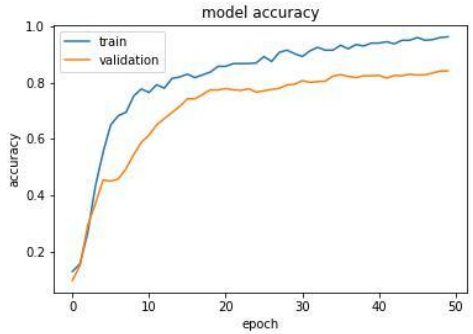
3. Add dropout layers to reduce overfitting.

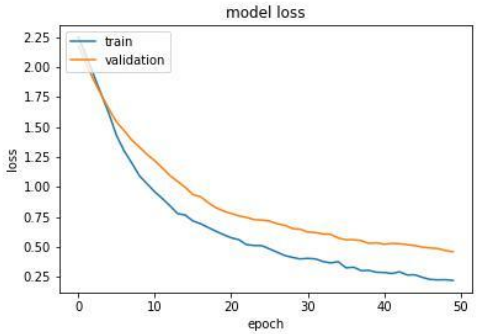
TASK 4: Improve learning curve: learning rate

LR:	0.00001	0.0001	0.001	0.01
				
				
Val. Acc.:	0.647	0.841	0.837	0.821.

TASK 4: Improve learning curve: learning rate

LR:	0.0001



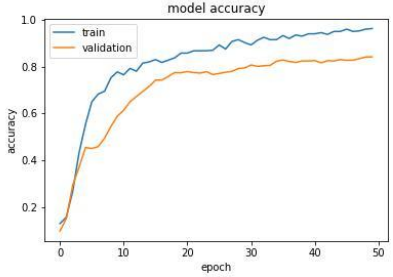


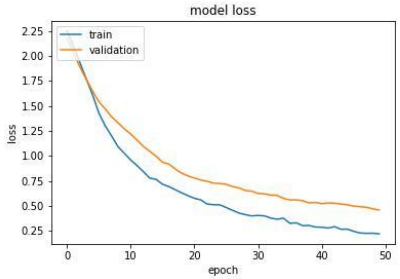
Val. Acc.:	0.841

With a smaller learning rate, we obtained much smoother learning curves and the results improved from 82% to 84%. Finding the right learning rate is important, as too small a learning rate can slow down training and too large a learning rate can cause oscillations. Therefore, we will use a value of 0.0001 as the learning rate in the next tasks.

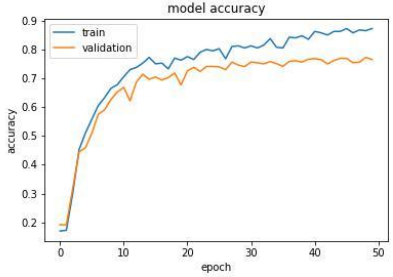
TASK 4: Improve learning curve: layer norm

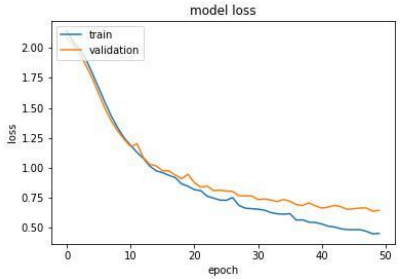
Batch Normalization





Layer Normalization





Val. Acc.:	0.841	0.773

Using Layer Normalizations, we obtained worse results than the original Batch Normalization layers. Therefore, we can conclude that a batch size of 32 is sufficient to obtain representative mean and standard deviation values.

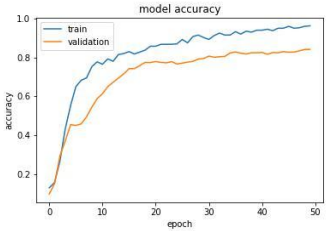
TASK 4: Improve learning curve: dropout

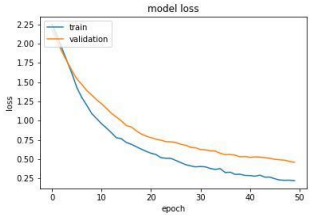
Probability

None

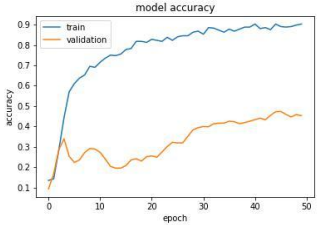
0.1

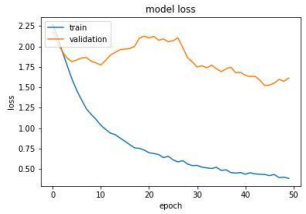
0.5



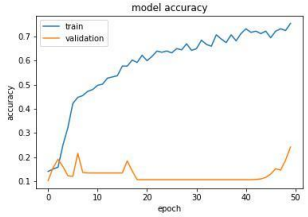


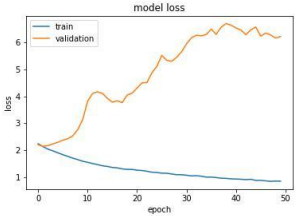
Val.: 0.841





Val.: 0.47





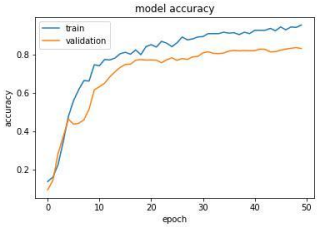
Val.:0.24

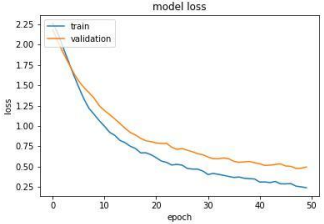
After every convolutional layer

Probability

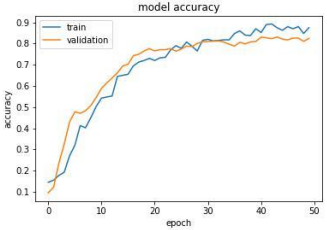
0.1

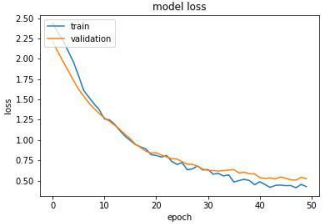
0.5





Val.: 0.837
Train: 0.95





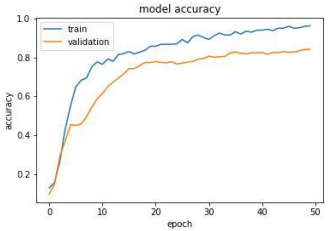
Val.: 0.831
Train: 0.893

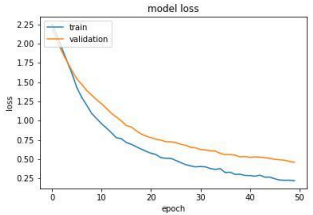
After last global average pooling

TASK 4: Improve learning curve: dropout

Probability

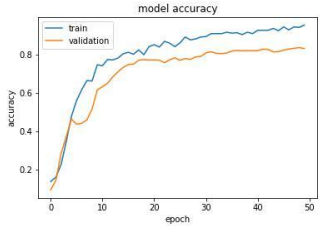
None

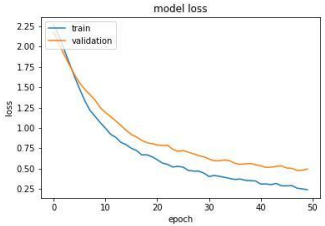




Val.: 0.841

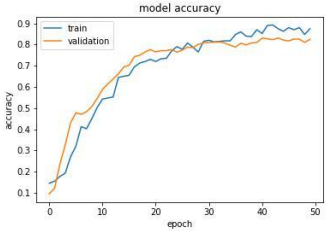
0.1

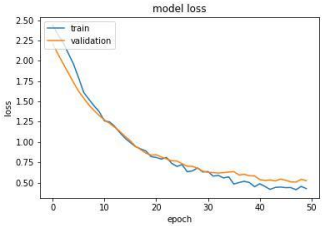




Val.: 0.837
Train: 0.95

0.5





Val.: 0.831
Train: 0.893

After last global average pooling

- Using dropout after every convolutional layer resulted in much worse results. This could be due to the conflict between the zeros created by dropout and the false mean and standard deviation calculated by batch normalization layers.
- Adding one single dropout layer after the last global average pooling (with no batch normalizations after), improved the overfitting, but the final results did not improve.
- We will not use dropout in the next tasks.

TASK 5: Hyperparameter search

Optuna was used to optimize our neural network, using the following parameters and ranges in a study :

Parameter	Range/Option
Batch size	[8, 16, 32, 64, 128]
Epoch	[10, 50, 100]
Optimizer	["SGD", "RMSprop", "Adagrad", "Adadelta", "Adam", "Adamax", "Nadam"]
Learning rate	[0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
Momentum	[0.0, 0.2, 0.4, 0.6, 0.8, 0.9]
Activations	["softmax", "softplus", "softsign", "relu", "tanh", "sigmoid",hard_sigmoid", "Linear"]

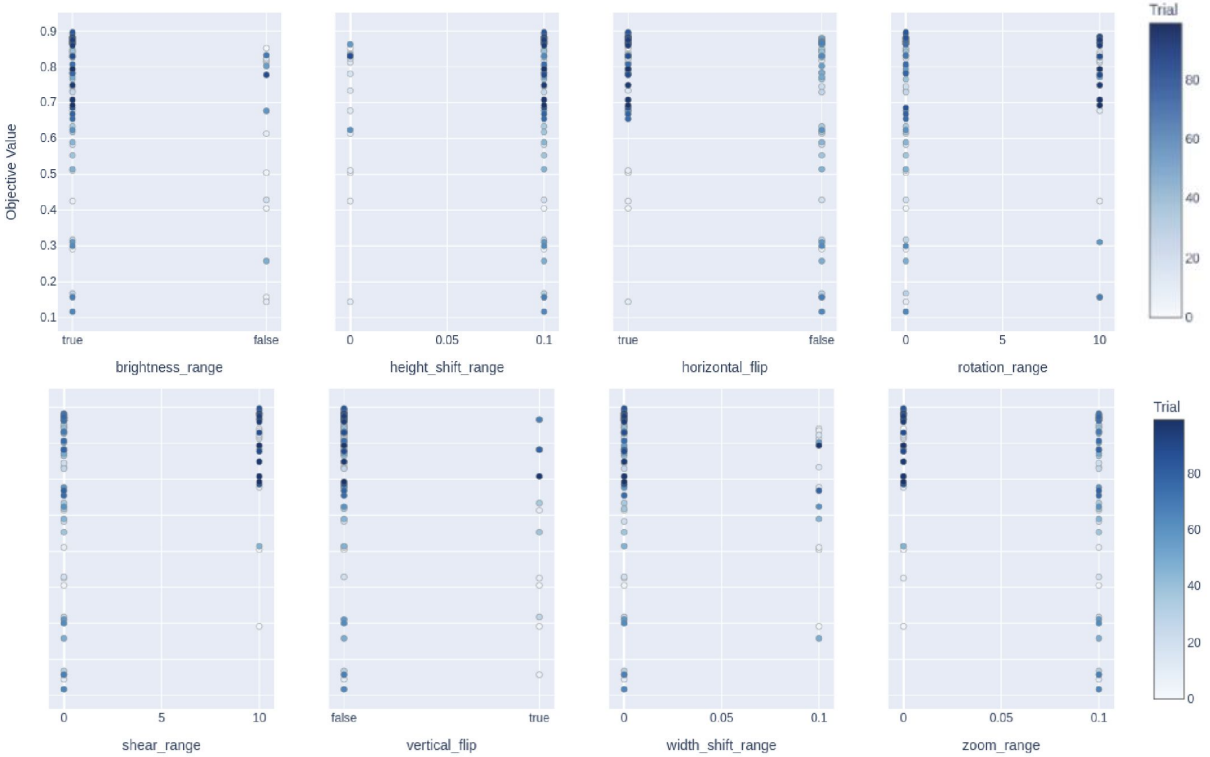
Data augmentation	Range
Width shift	[0.1, 0]
Height shift	[0.1, 0]
Horizontal flip	[True, False]
Vertical flip	[True, False]
Rotation	[10,0]
Brightness	[True, False]
Zoom	[0.1, 0]
Shear	[10, 0][

TASK 5: Hyperparameter search

Best score: 0.89

Results: trial winner parameters of data augmentation

Data augmentation	Winner
Width shift	0
Height shift	0.1
Horizontal flip	True
Vertical flip	False
Rotation	0
Brightness	True
Zoom	0
Shear	10



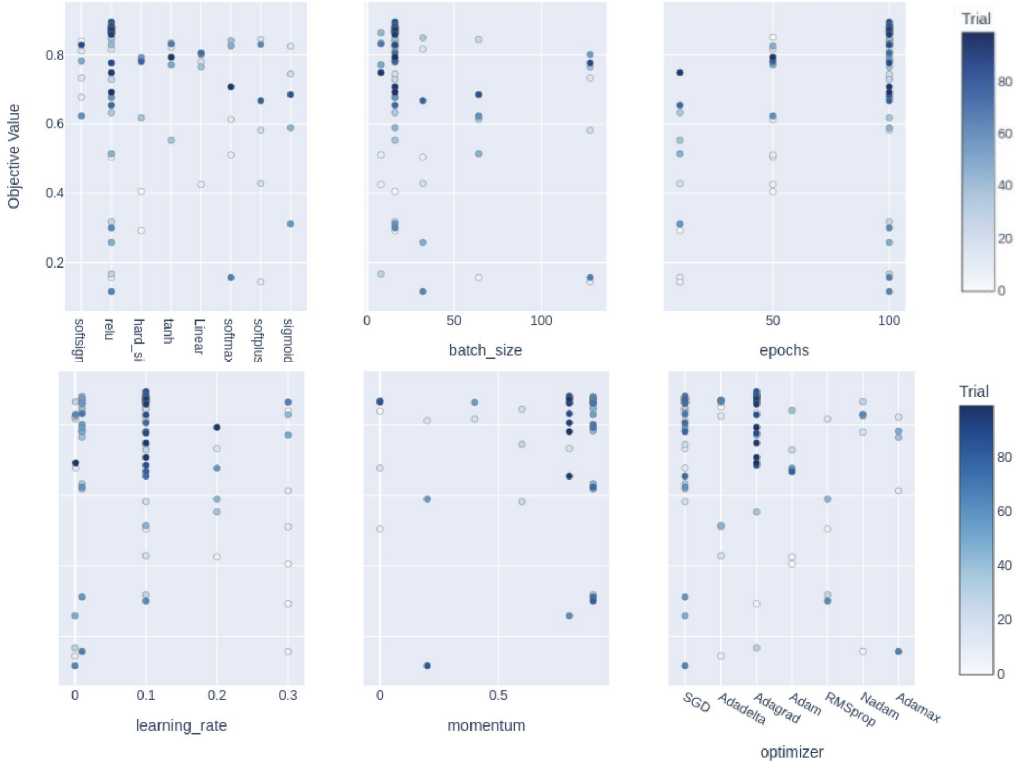
TASK 5: Hyperparameter search

Results: trial winner parameters

Best score: 0.89

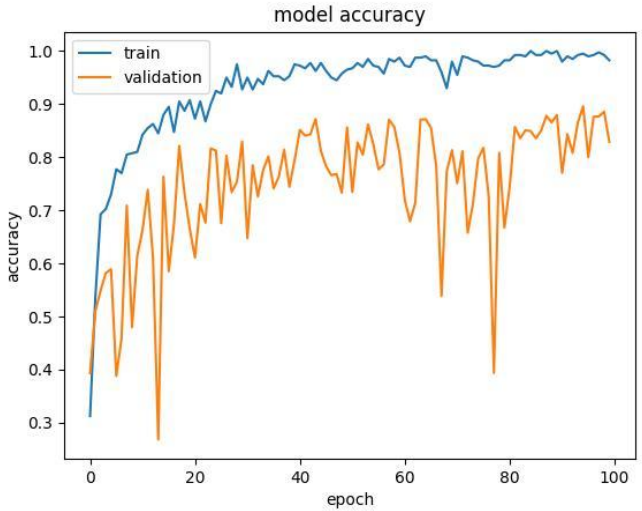
Parameter	Winner
Batch size	16
Epoch	100
Optimizer	Adagrad
Learning rate	0.1
Activations	Relu
Epochs	100

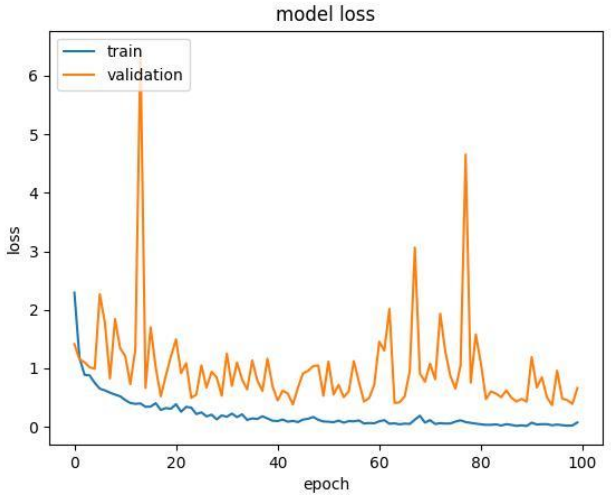
Optuna	
Trials	100



TASK 5: Hyperparameter search

Results: Accuracy and Loss graphs

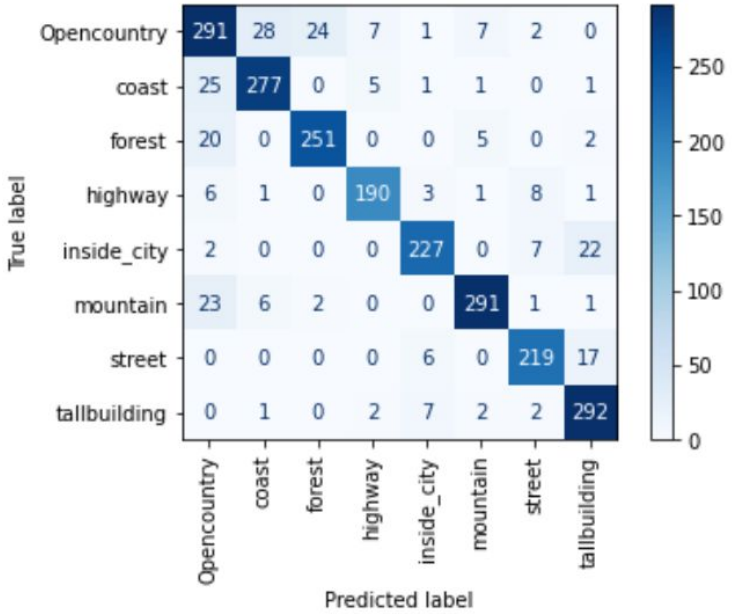


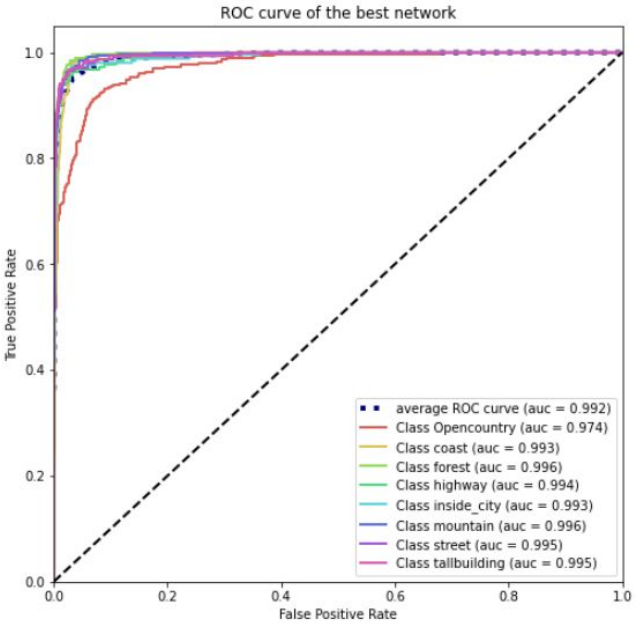


Best score: 0.89

Loss: 0.3685

TASK 5: Hyperparameter search





Best score: 0.89

The lowest results were obtained for images belonging to the Opencountry category, as they were frequently misclassified as coast, forest, or mountains. These images can be similar, making classification difficult.

The results for the other categories are very good.

Conclusions

- CNNs perform better than MLPs for image classification, as we achieved better results than in previous weeks.
- Training a large network is challenging when there is limited training data.
- We achieved a validation accuracy of 89% with a network having 500k parameters (108 times less parameters than the original network) using only 400 training images.
- It is important to keep in mind that a neural network design is an iterative process, and it may take several iterations to achieve the desired level of performance. Additionally, it is important to evaluate the model on a diverse set of data, to ensure that the model generalizes well and performs well on unseen data.
- Hyperparameter search for neural networks is a crucial step in training and fine-tuning a model for a specific task, this process can be time-consuming and computationally expensive, but it is essential for achieving high accuracy and generalization in the model. Techniques such as grid search and random search can be used to efficiently search the hyperparameter space.

Summary:

- New model
  - -Method selected: #2 (Block35-s)
  - -parameters: 567896
  - -Large dataset: val_accuracy**:** 0.9
  - -Small dataset: val_accuracy: 0.75, 0.76 0.76 0.779 (small training #1, #2, #3, #4)
- Data augmentation
  - - Technique with best results: Shear transformation (No improvement mixing techniques)
  - - Improvement: From 0.75% to 0.82 (validation accuracy)
- Learning rate:
  - - Original learning rate: 0.01
  - - Best learning rate results: 0.0001
  - - Improvement: from 0.82 to 0.84 (validation accuracy)
- Hyperparameter search
  - -Batch size: 16 -Epochs: 100
  - -Optimizer: Adagrad -Learning rate: 0.1
  - -Number of trials: 100 - Data augmentation: W_s: 0 H_s: 0.1 Zoom: 0 Horizontal Flip: True
  - -Activation: Relu Vertical Flip: False Rotation: 0 Shear: 10 Brightness: True
- Best Results with new model (Small Dataset):

Best score: 0.89