Fruits and vegetable image recognition using convolutional neural networks

Biologically inspired artificial intelligence

Aleksander Świtała

Informatics
Silesian University of Technology
30.06.2022

1 Introduction

The main aim of the project is to recognize fruits and vegetables provided by the database. This is the classic project topic used for presenting the capabilities of processing images using convolutional neural networks.

The main tasks of the algorithm will be to recognize using detection if there are any fruits/vegetables from the database spotted on the image and label them using appropriate names thanks to classification.

2 Analysis of the task

2.1 Convolutional neural networks

Convolutional neural networks belong to deep neural networks and can gradually filter different parts of the training data and sharpen important features in the discrimination process used to recognize or classify patterns. This type of neural network is predestined for computations on 2D structures.

The main part of these networks is a layer that uses an operation called convolution. Convolutions allow for the extraction of simple features in the initial layers of the network, e.g. they recognize edges with different orientations or stains of different colors, and then slices in subsequent layer:

- **Input Layer** - Inputs images to the network and applies data normalization.
- **Convolutional Layer** - Convolutional layers contain learned filters that are used to distinguish images from each other. Parameters:
    - Kernel/Filter Size - refers to the size that is moved over the window input.
    - Padding - defines how the sample size is handled, which makes it possible to obtain the same size of the output as the size of the input.
    - Strides - indicates by how many pixels the filter window should be moved.
    - ReLU (Rectified Linear Activation) - uses non-linearity in the model. It is very quick to train.
- **Pooling Layers** - Uses downsampling on neighboring pixels with similar values to reduce the size of the input, lowering the number of parameters to train, shortening the time of network working, and simplifying the model.
- **Batch Normalizing Layer** - Normalization is the most effective way to fight network overtraining. It standardizes data making the process of training smoother, and faster.
- **Flatten Layer** - Flatten layer transforms the multidimensional network layer into a one - dimensional vector to match the input to the classification. The most commonly used function is SoftMax to classification of these features, which requires univariate data.

2.2 Dataset

There are a lot of publicly available datasets on websites such as kaggle.com, roboflow.com, and data.world.

The used dataset was from the Kaggle website and consists of images of 30 types of fruits and vegetables bulk imported from Google Gallery. There was a need to prepare the dataset before training, as there were different formats than .jpg, images of different dimensions, and channels size, also the validation and test datasets were the same so some modification was needed.

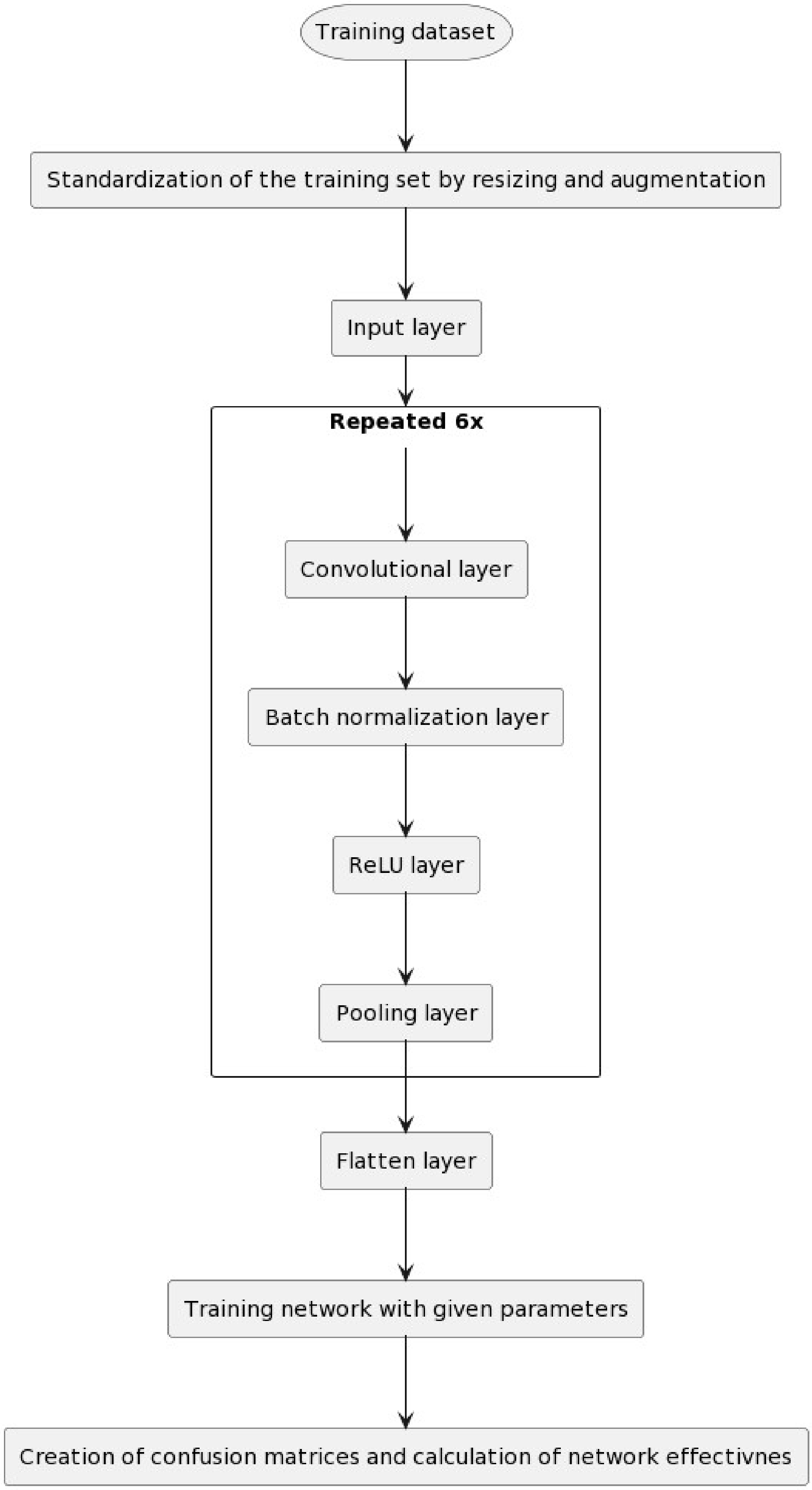
2.3 Tools

In most cases, the python with TensorFlow and Keras or Pandas libraries stack is used for the implementation of image recognition and AI projects as a whole.

In my case, I used the MATLAB environment as it has all the needed libraries and tools already preinstalled, and has an app designer helping develop the graphical user interface. The MATLAB environment helps in debugging the code, and autocorrect with redirects to documentation containing all the necessary information required of the searched function with extensive description and examples, helping in faster development and training of the cnn.

3 Internal and external specification

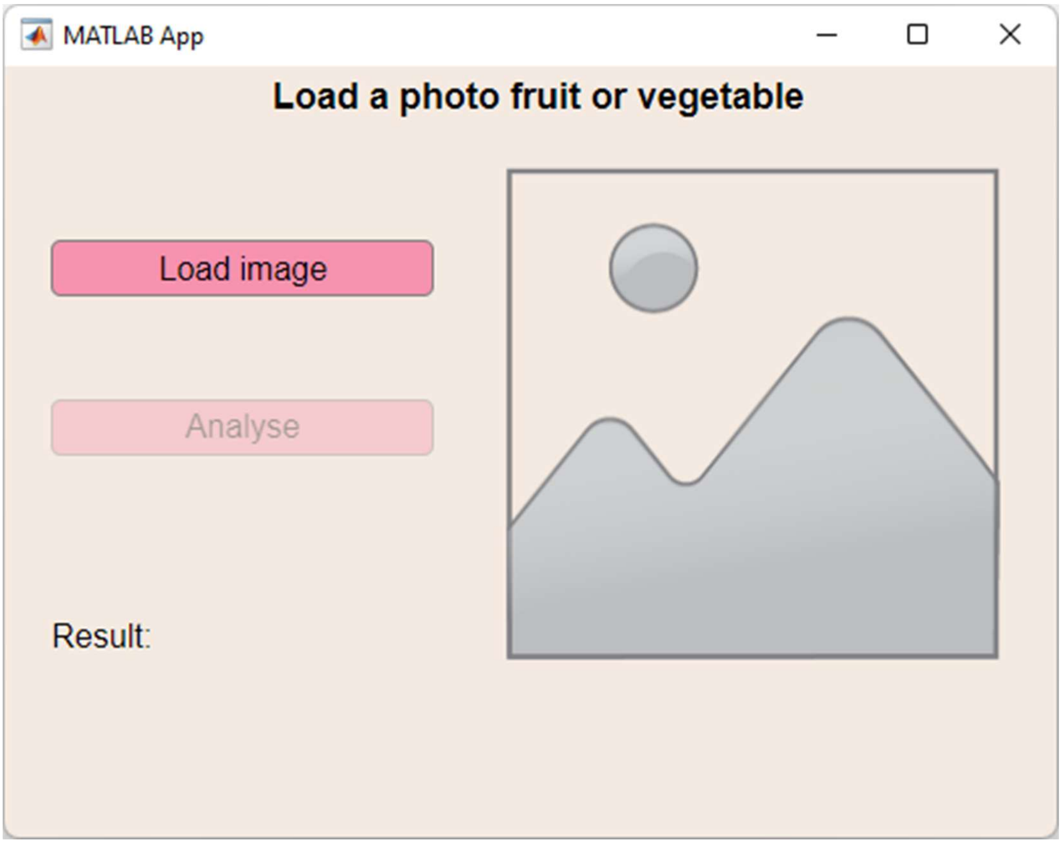
3.1 CNN training schema



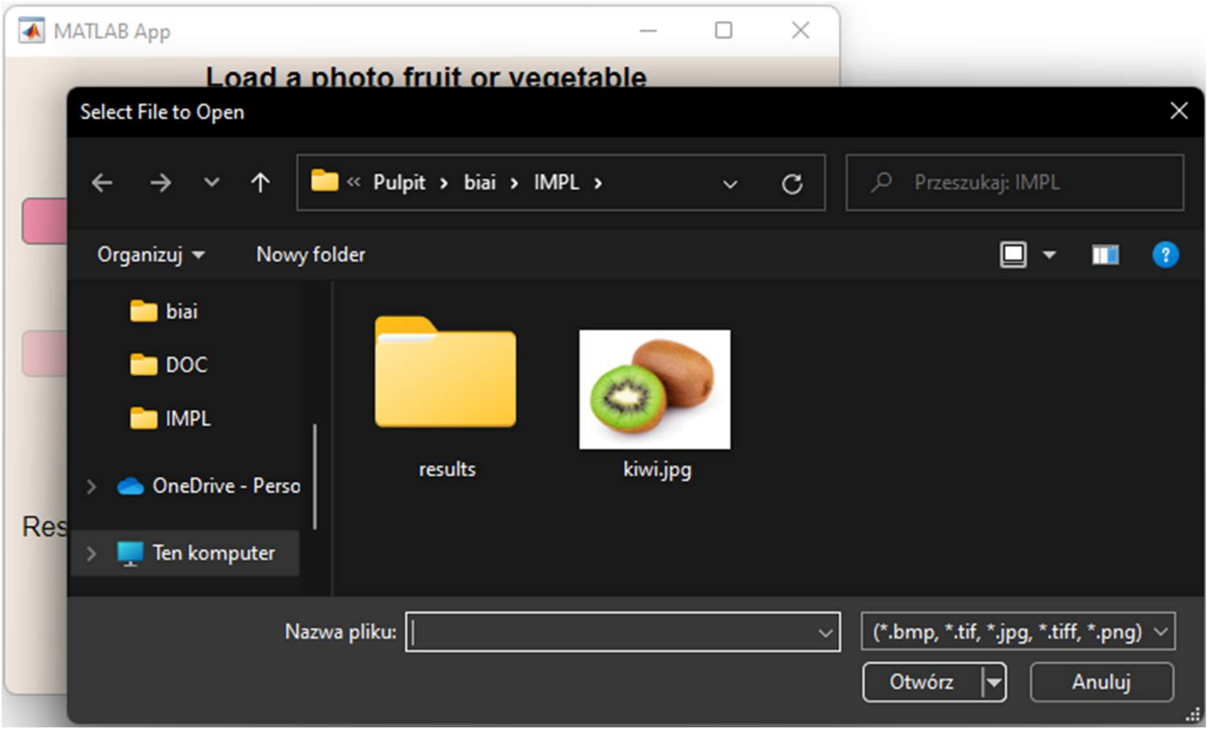
3.2 Data structures

	Train Set	Validation Set	Test set
Apple	499	49	50
Banana	540	65	65
Broccoli	496	50	50
Carrots	496	50	50
Cauliflower	498	50	50
Chili	395	50	50
Coconut	500	50	50
Cucumber	498	50	50
Custard Apple	500	50	50
Dates	496	50	50
Dragon	500	50	50
Egg	500	50	50
Garlic	500	50	50
Grape	499	50	50
Green Lemon	499	50	50
Jackfruit	500	50	50
Kiwi	497	50	50
Mango	500	50	50
Okra	500	50	50
Onion	500	50	50
Orange	496	48	48
Papaya	500	50	50
Peanut	500	50	50
Pineapple	500	50	50
Pomegranate	499	50	50
Star Fruit	496	50	50
Strawberry	500	50	50
Sweet Potato	499	50	50
Watermelon	497	50	50
White Mushroom	498	50	50
Together	14898	1512	1513

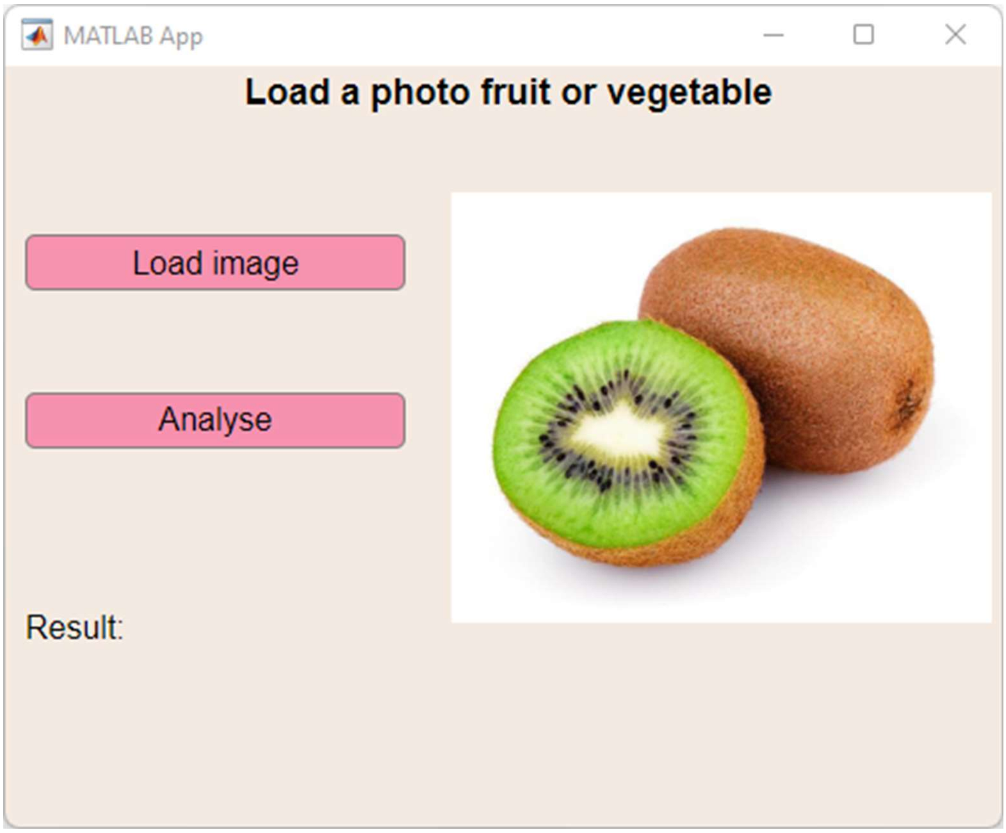
3.3 GUI



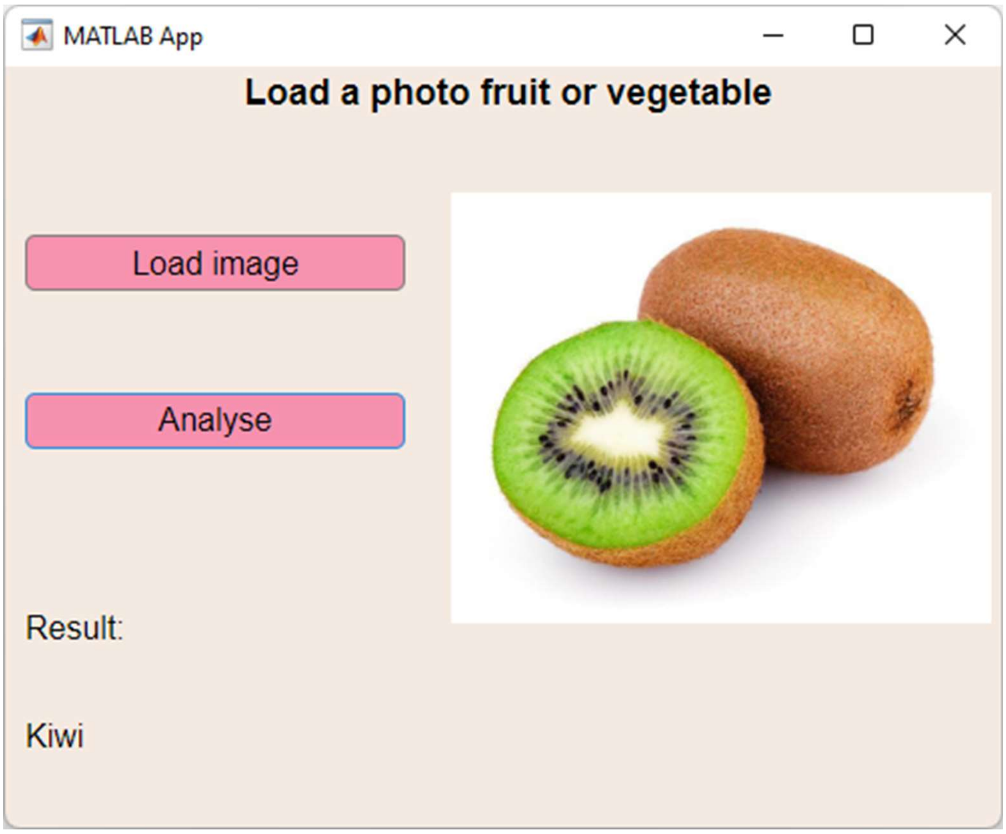
Start menu



Choosing an image to load



Loaded image



Predicted fruit/vegetable

4 Experiments

4.1 Background

The initial parameters of the network:

- ImageDataAugmenter – RandRotation [-20,20], RandXTranslation [-3,3],RandYTranslation [-3,3]
- Convolutional2dLater number of filters – 4/8/16
- Convolutional2dLater size of filters – 3
- TrainingOptions - SolverName adam, LearnRateDropFactor 0.8, LearnRateDropPeriod 5, ValidationPatience 3, OutputNetwork last-iteration

Parameters subject to changes and testing:

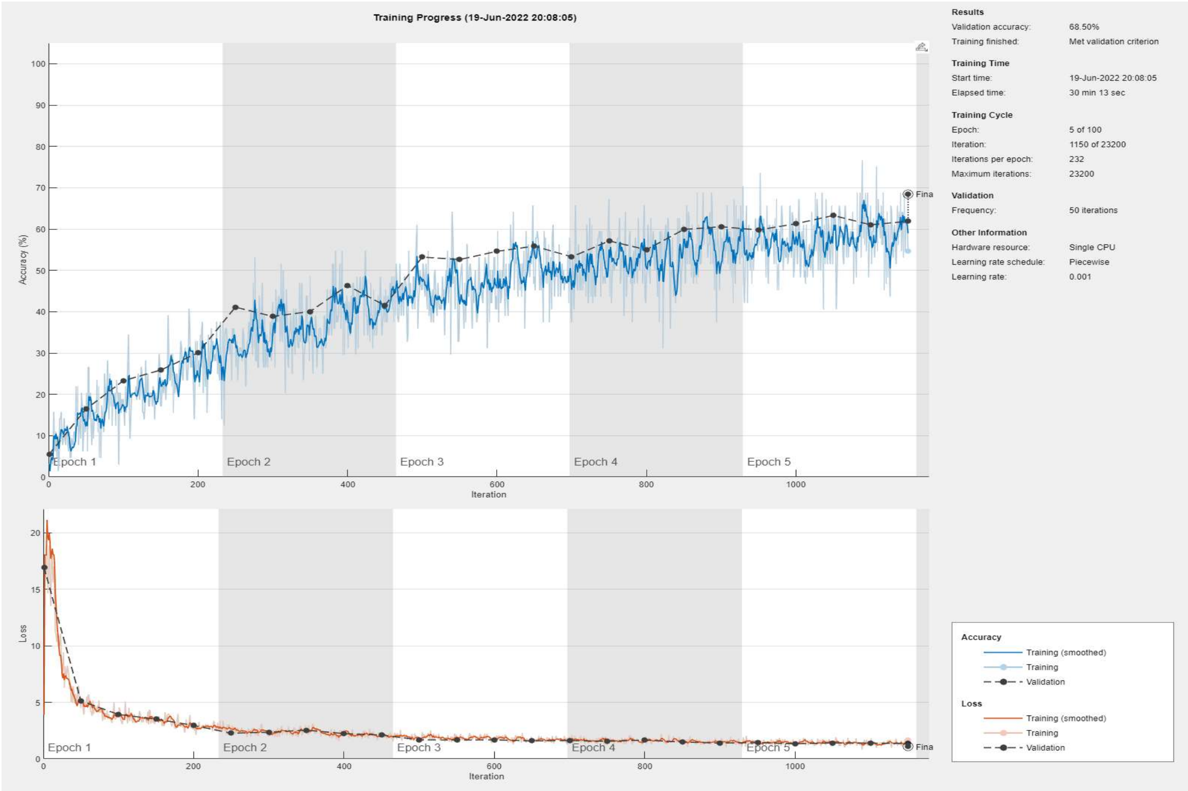
- Number of filters – increasing the number of filters usually increases the accuracy of the training, and the amount of time needed for the training.
- SolverName – there are 3 options with different optimizers which are Adam, Sgdm, and Rmsprop.

The results were presented on a plot with extensive descriptions, and matrices of test/training/validation errors.

4.2 Performed tests

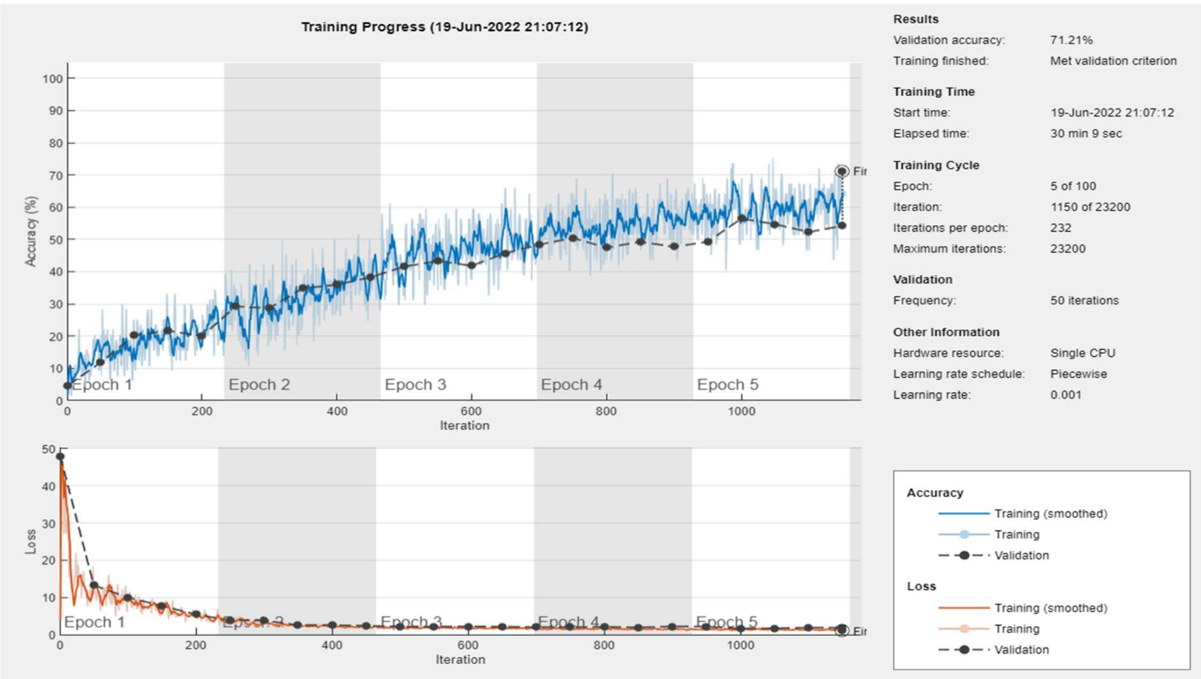
4.2.1 Number of filters – 3, SolverName – Adam

Initial training parameters. A benchmark for later tests.



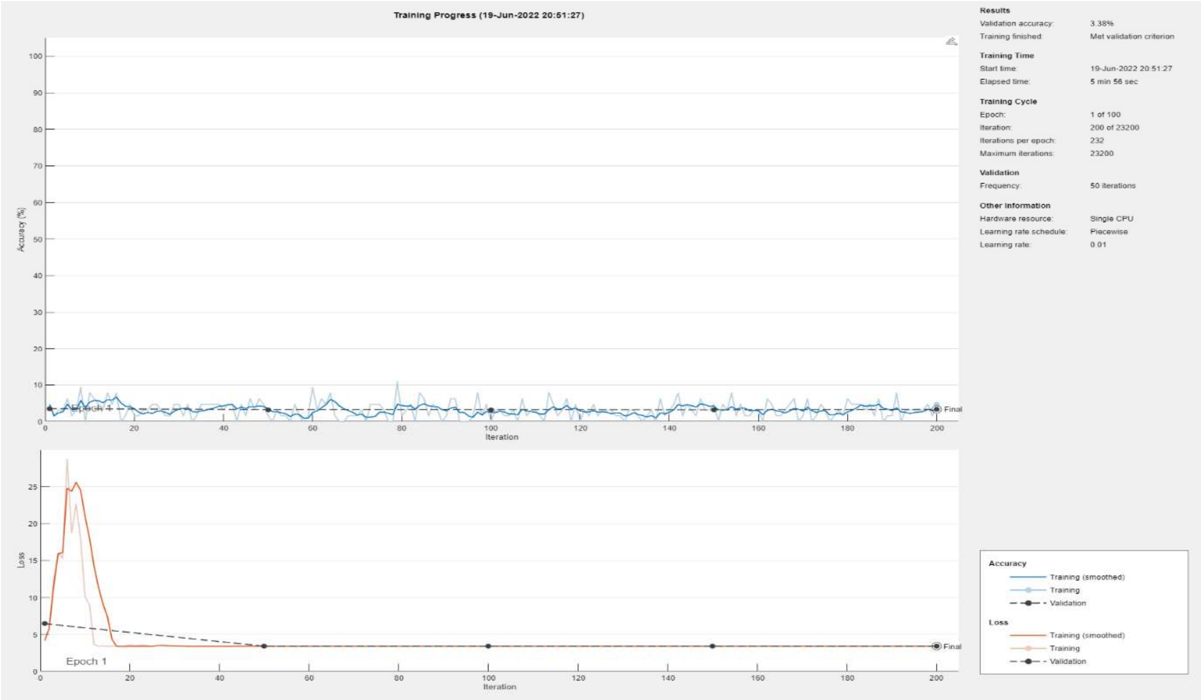
4.2.2 Number of filters – 3, SolverName – Rmsprop

Changing the SolverName to **Rmsprop** resulted in increase of validation accuracy and the training time didn't increase.



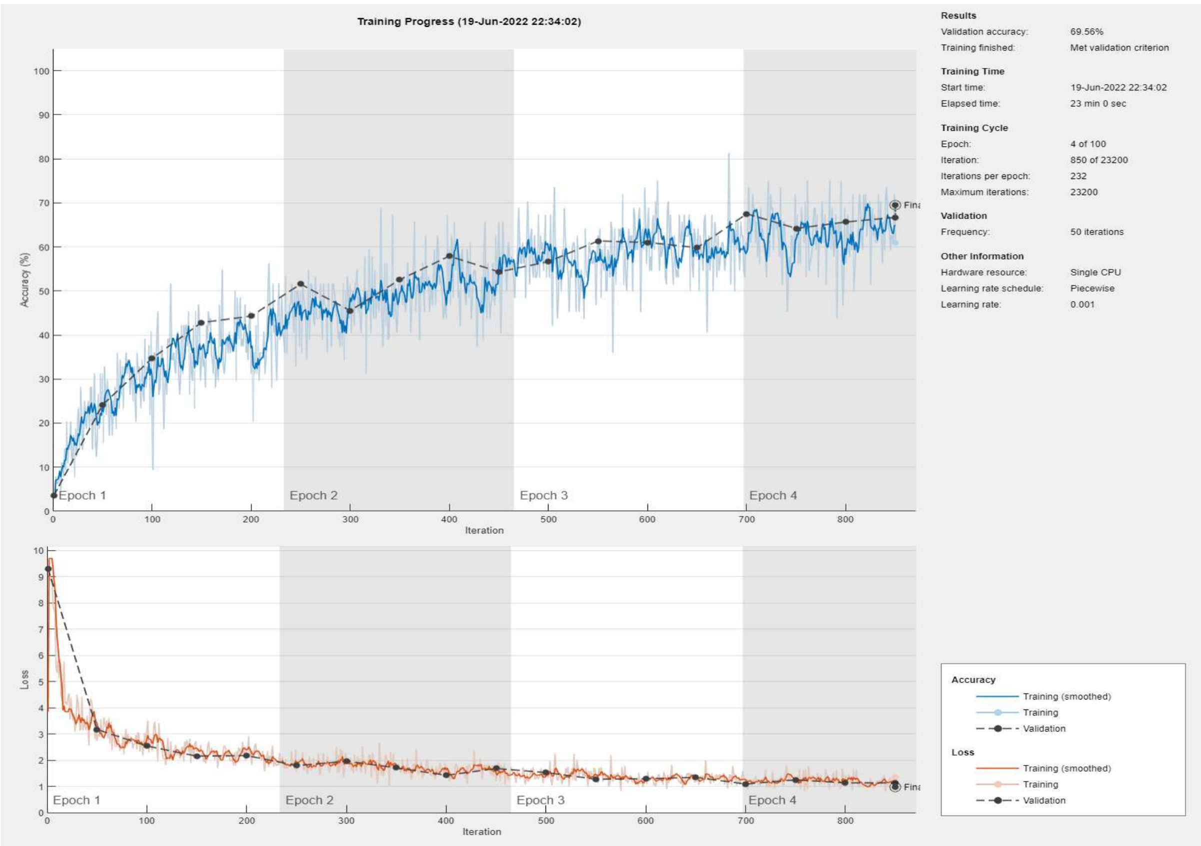
4.2.3 Number of filters – 3, SolverName – Sgmd

The training of **Sgmd** SolverName was canceled as the result were unsatisfactory. I decided to drop the training of SolverName.



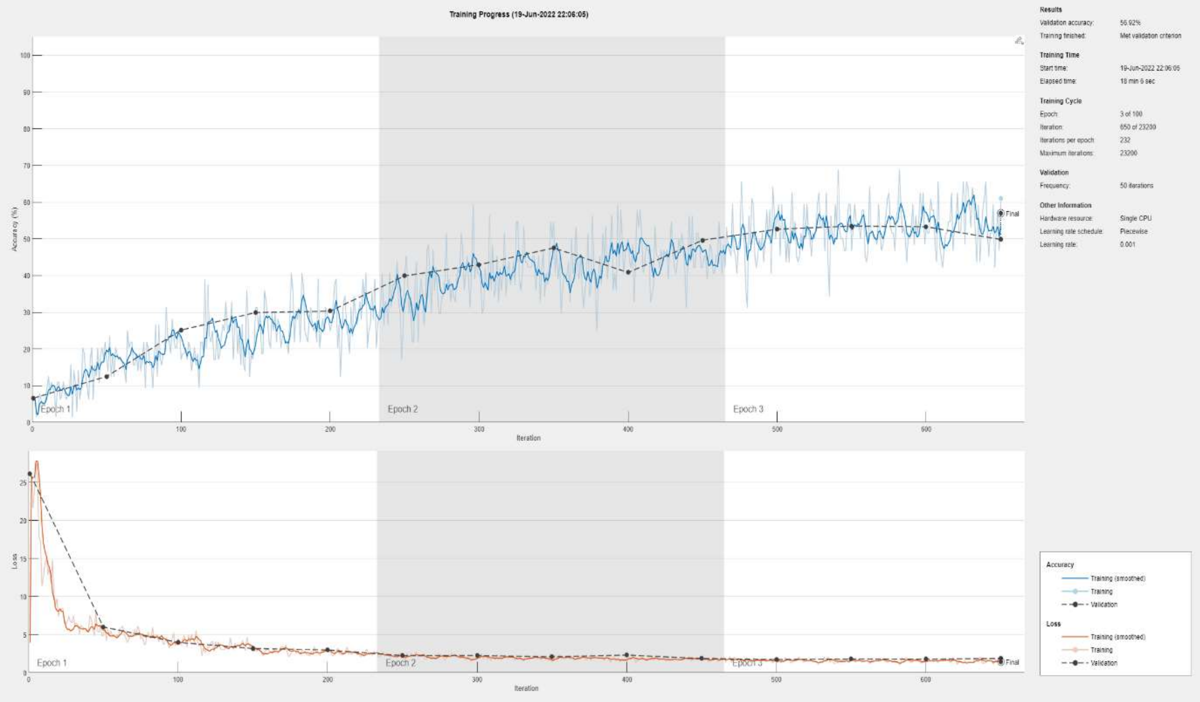
4.2.4 Number of filters – 4, SolverName – AdaAdam

The accuracy increased compared to the 3 No. filters with the same SolverName, but was still worse than the one with SolverName **Rmsprop**. Interestingly the time of training decreased.



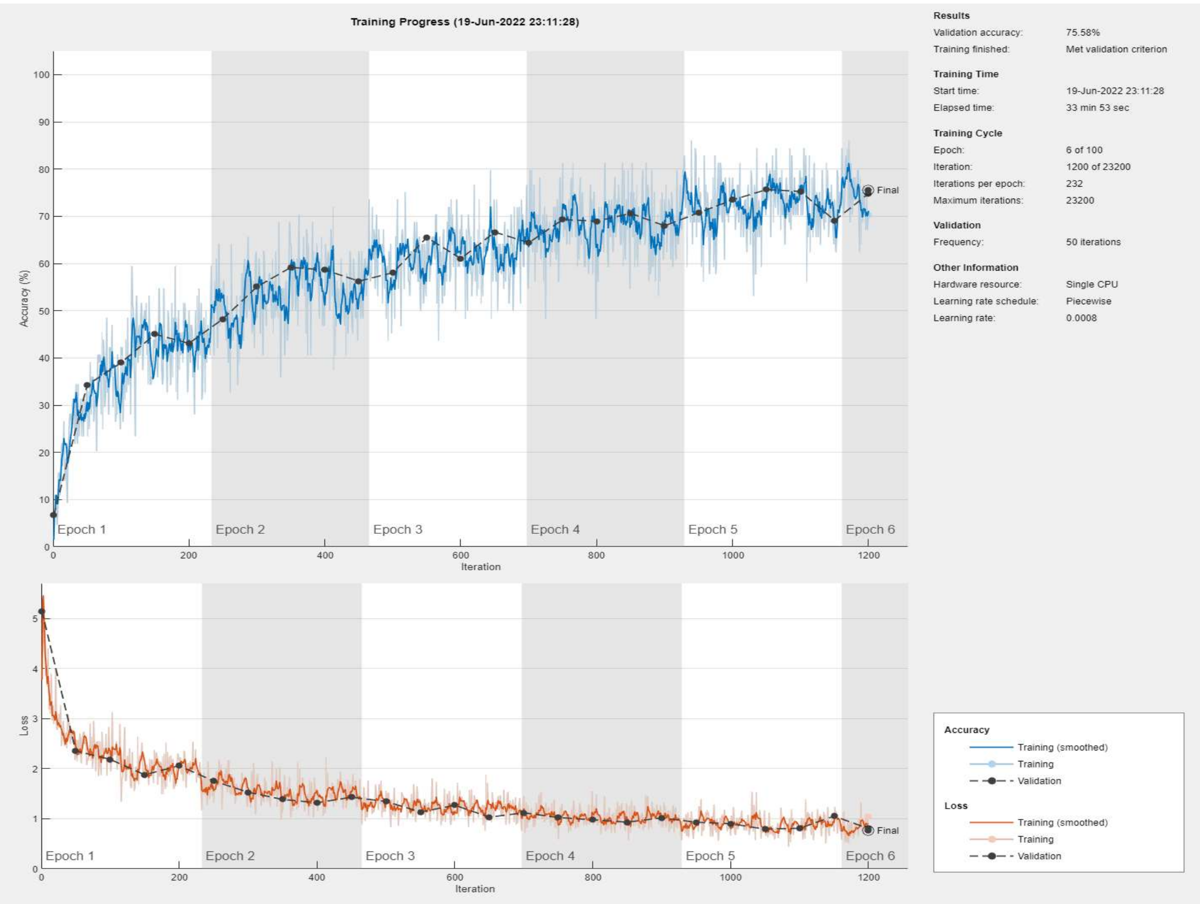
4.2.5 Number of filters – 4, SolverName – Rmsprop

The time and accuracy largely decreased after increasing No. filters for **Rmsprop**.



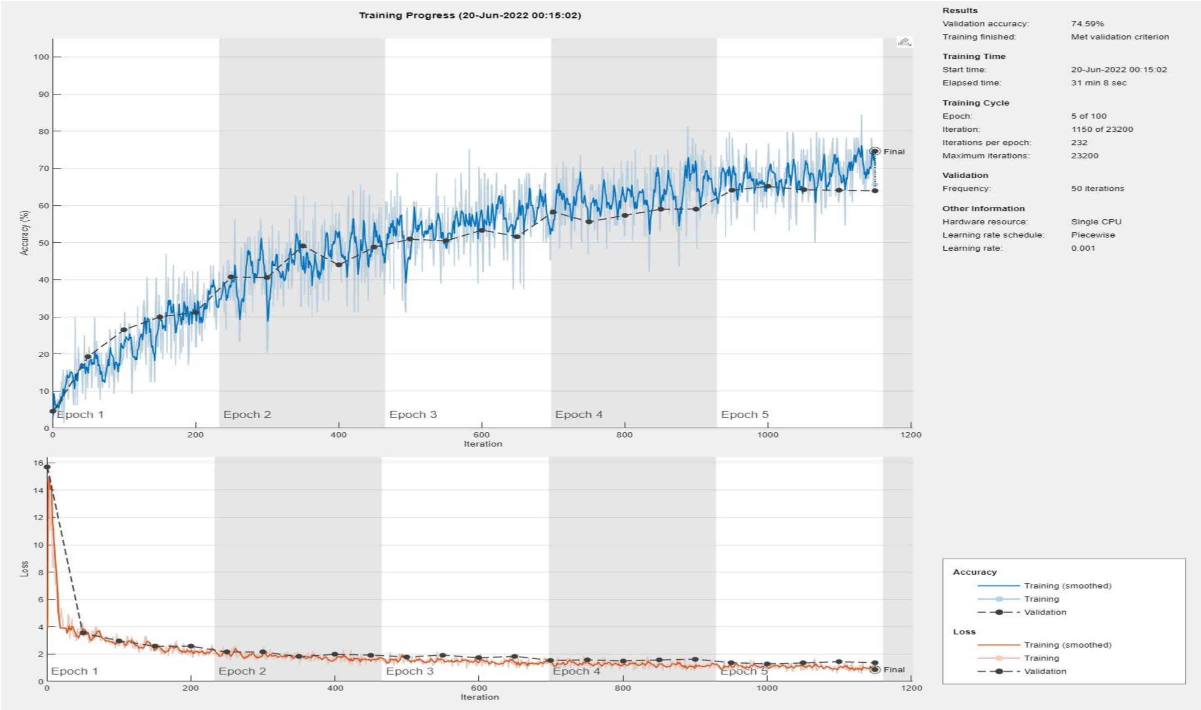
4.2.6 Number of filters – 5, SolverName – Adam

The time and accuracy, in comparison to the initial network, increased by 10% (respectively 3 minutes and 7 percentage points).



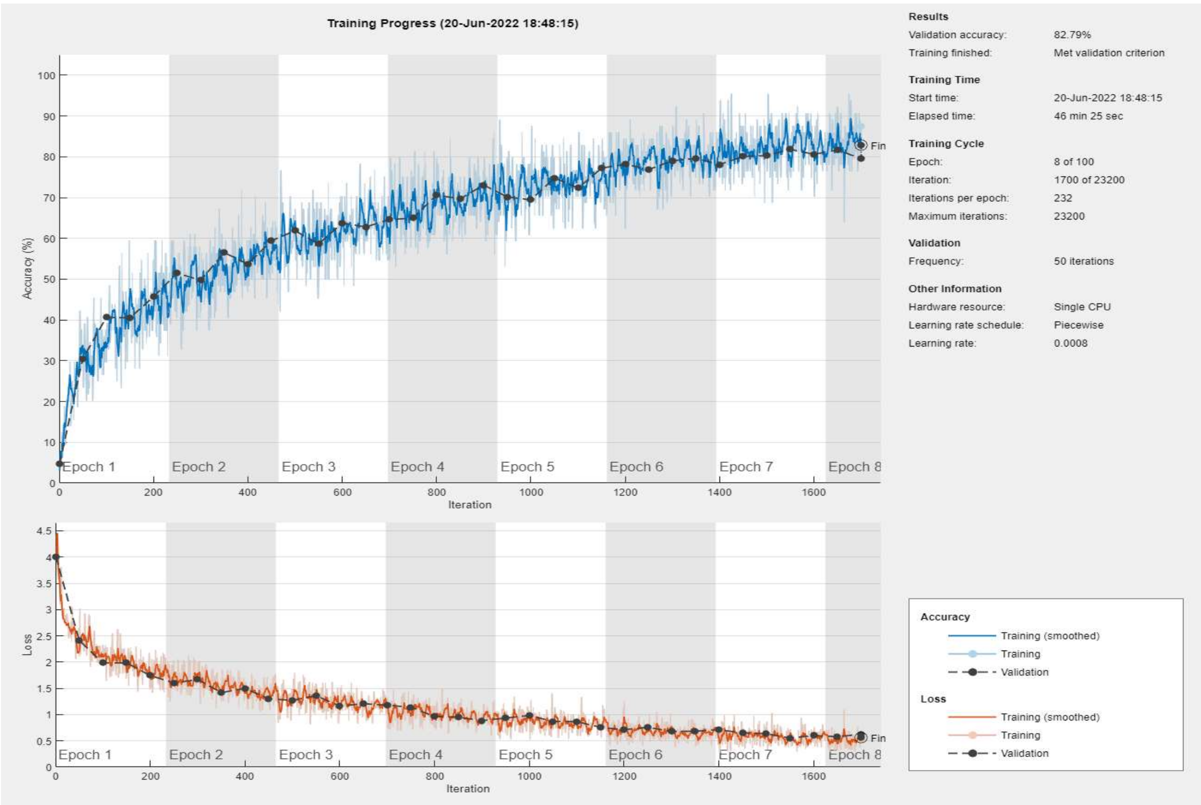
4.2.7 Number of filters – 5, SolverName – Rmsprop

The accuracy was the same as for the network with the same No. of filters and with the SolverName **Adam**, but the time decreased by 2 minutes.



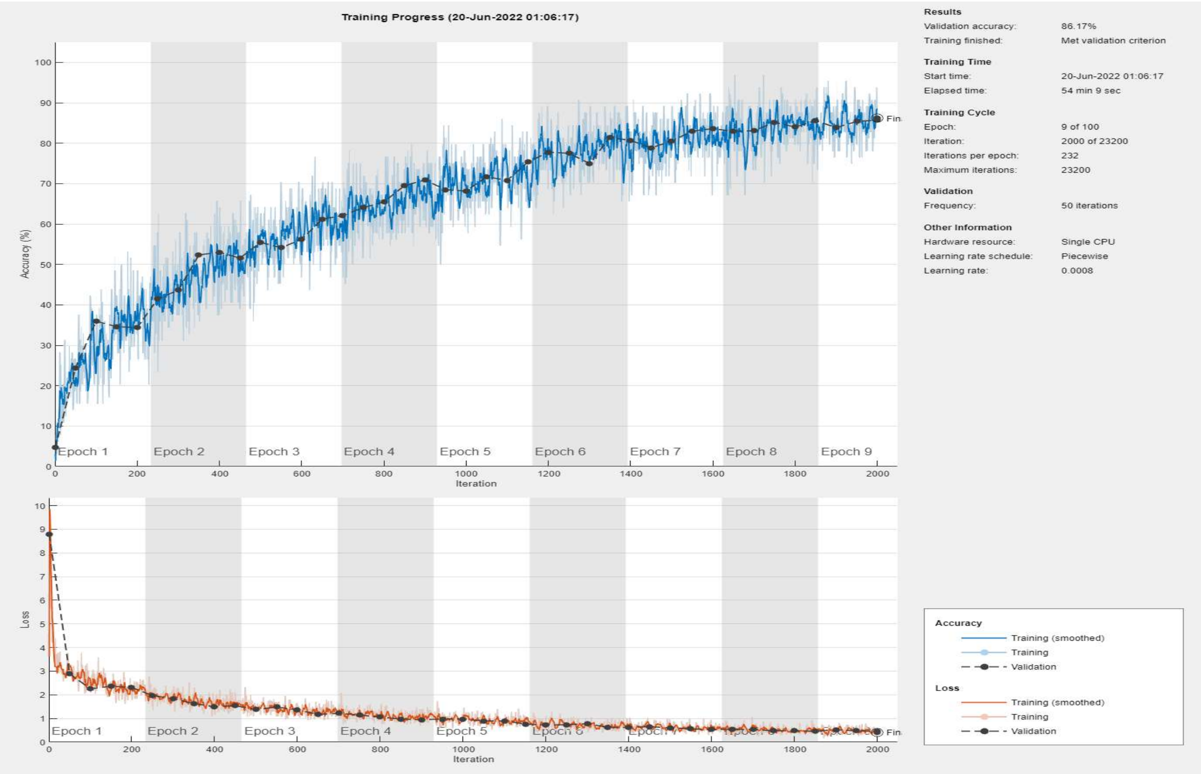
4.2.8 Number of filters – 6, SolverName – Adam

The accuracy finally reached a mark above 80%, but the time was largely increased to 46 minutes.



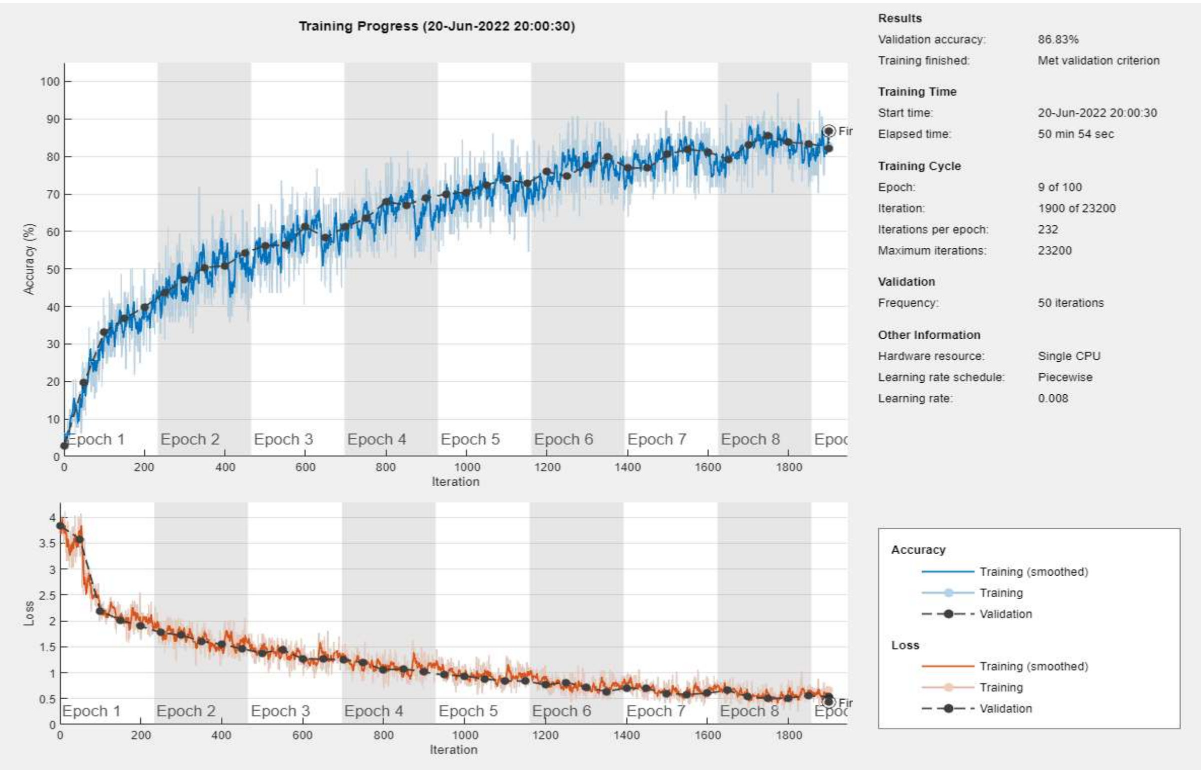
4.2.9 Number of filters – 6, SolverName – Rmsprop

Compared to previous network accuracy increased by 4 percentage points and time by 8 minutes.

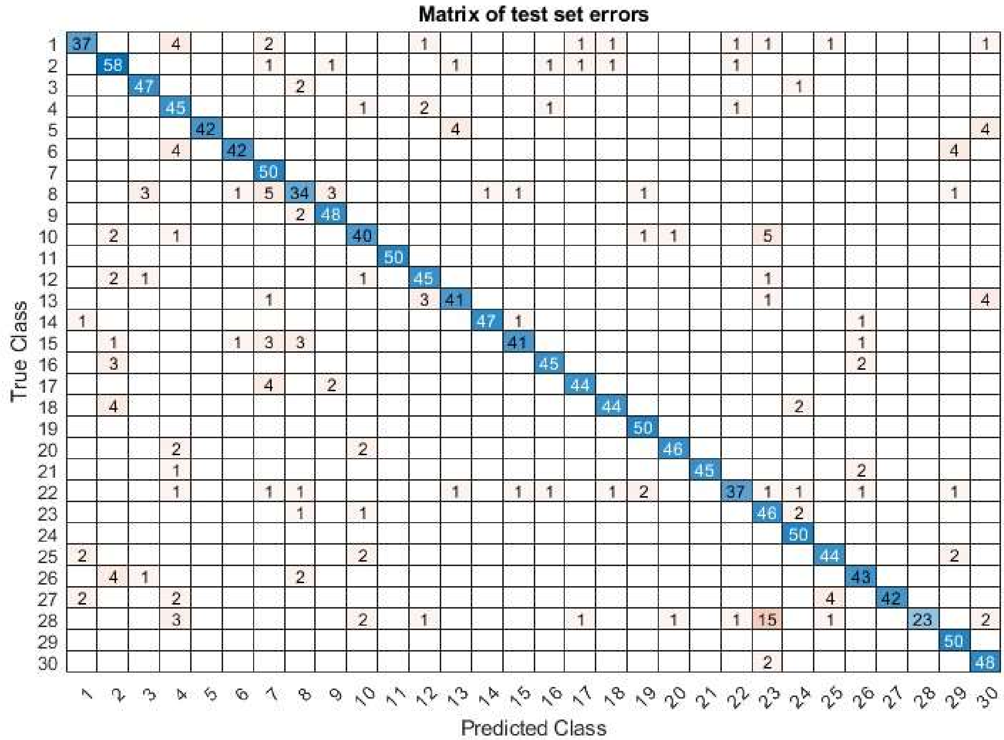


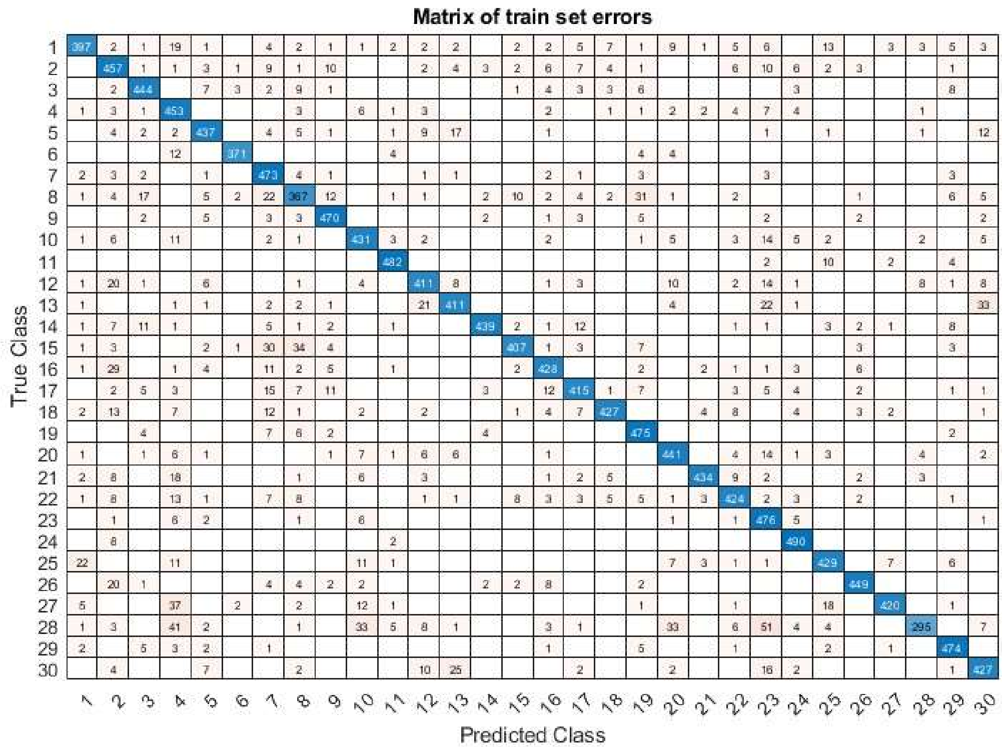
4.2.10 Number of filters – 6, SolverName – Sgmd

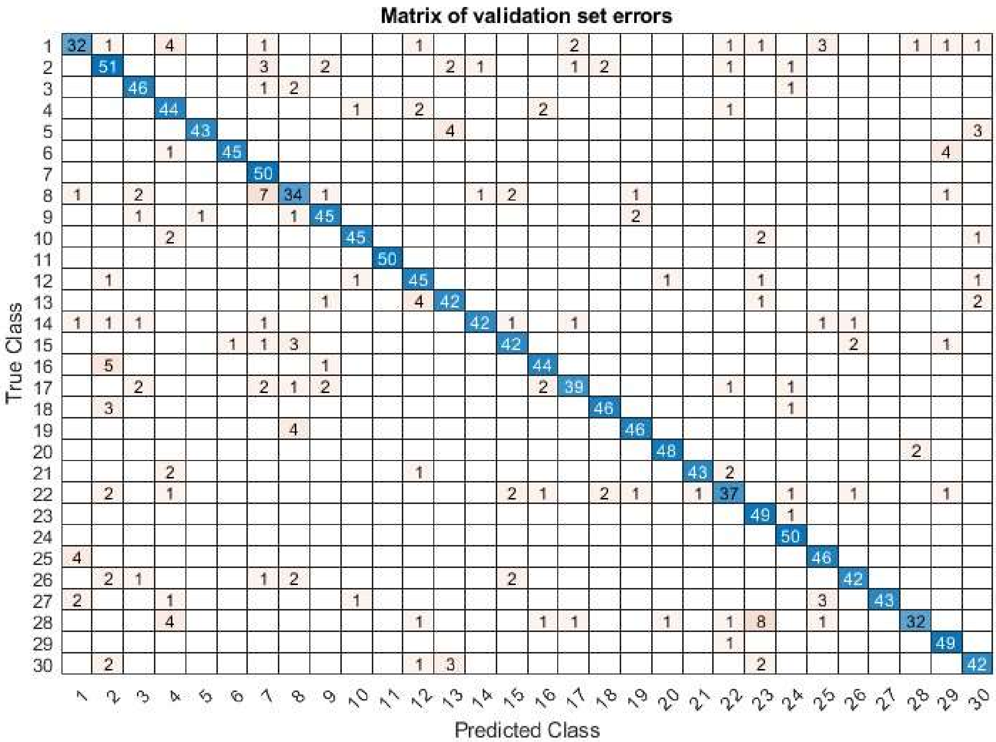
At the last training, I decided to also test **Sgmd** SolverName if it could perform better than on the first try. The test surprised me, as the trained network reached the same result as currently the best one (the previous network), but with time decreased by 4 minutes resulting in the best network.



Matrices of test, training, and validation set errors of the best trained network:







5 Conclusions

The development and training of convolutional neural networks don't seem as difficult as I thought before. It was fun to try to develop some kind of AI, which as a branch of computer science is currently one of the most exciting topics in new technologies and inventions, which will revolutionize the world. In the future to improve my network I could create my dataset with more differentiation of details of current fruits or vegetables.

6 References

- https://miroslawmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/
- https://www.mathworks.com/discovery/image-recognition-matlab.html
- https://www.fritz.ai/image-recognition/
- Dataset - https://www.kaggle.com/datasets/shadikfaysal/fruit-and-vegetables-ssm

7 Links

- GitHub - https://github.com/alekswi/biai