IceCore MeltLayer Predictions using a CNN

Advanced Machine Learning

Alkis Zoumis (snk737)
Eirini Malegiannaki (sbj276)
Julius Seumer (hwg245)
Filippos Ktistakis (fwb590)
Rasmus Borup (phr418)

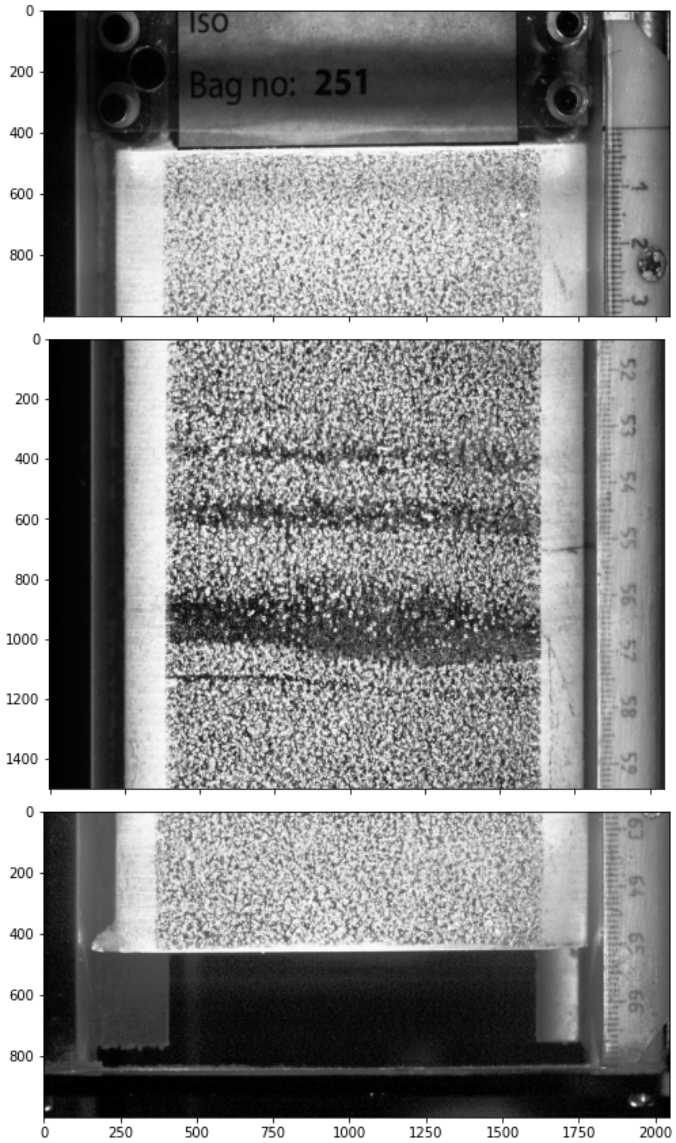
June 14, 2022
Niels Bohr Institute

Introduction

Ice Cores from Antarctica and Greenland are unique paleo climate archives

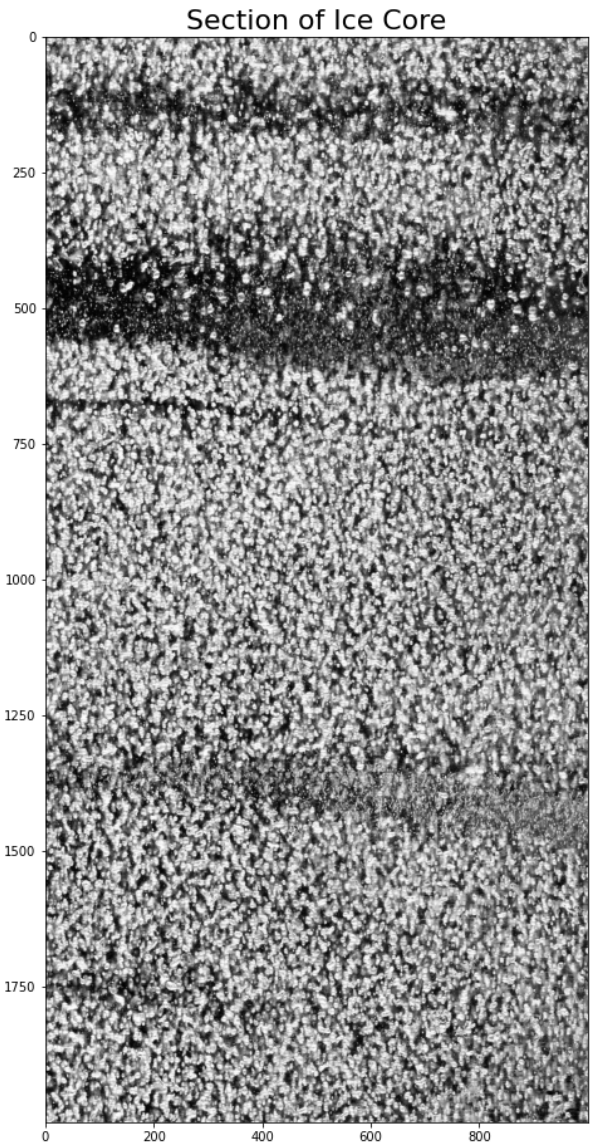
Ice cores in Antarctica go back in time up to 800.000 years and the Oldest Ice Core that will cover a climate history of 1.5 million years is being drilled now.

The 'Data Set'



- 2105 m long Ice Core from Greenland
- Cut in to 1.65 m long sections
  - more than 1200 pictures
  - not all with the same camera settings (focus, illumination, etc.)
  - we select specific setting –> 210 images
- **General Idea:** slice image in smaller parts and predict if slice contains melt layer or not
- **Problem:** Mapping labels on to image

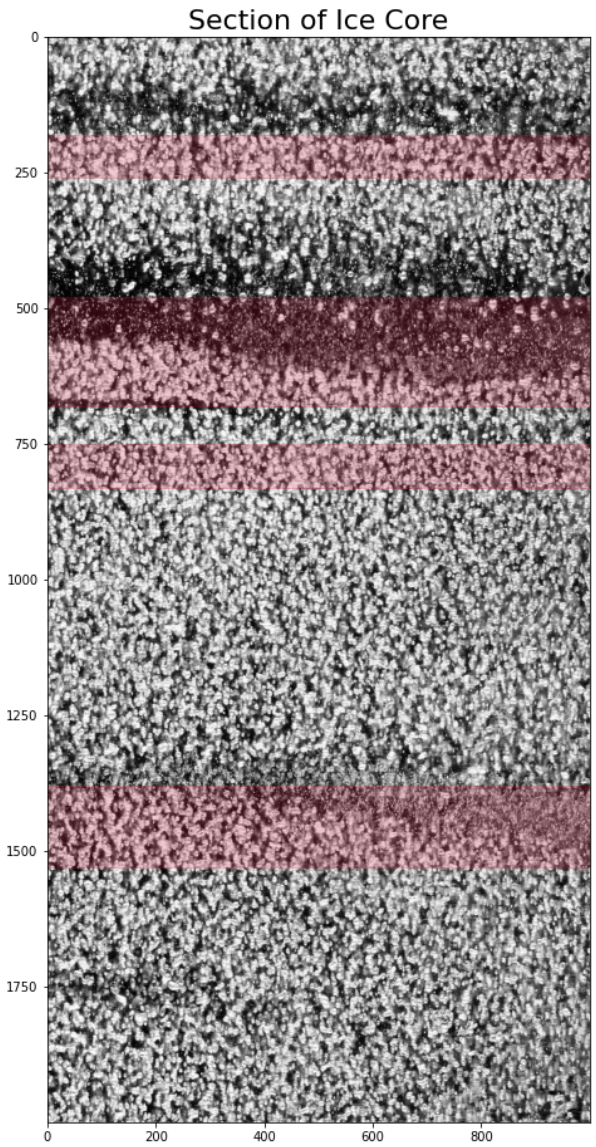
Preprocessing



bag	depth from	depth to	age (b2k)	age (CE)	type of feature
218	120.5650	120.565	858.045	1142.0	meltLens
NaN	121.7200	NaN	868.236	1131.8	meltLayer uncertain
221	122.3063	122.3063	872.875	1127.1	meltLens
NaN	122.3900	NaN	873.636	1126.4	meltLens uncertain
NaN	123.5600	NaN	885.927	1114.1	indication of bubble free layer or patch
224	124.0487	124.0487	891.259	1108.7	meltLens
224	124.2134	124.2134	893.055	1106.9	meltLens
227	124.7216	124.7216	897.066	1102.9	meltLens
227	125.7708	125.7745	905.697	1094.3	meltLayer
230	127.2374	127.2374	917.704	1082.3	meltLens
233	128.1094	128.1094	925.631	1074.4	meltLens

- Table with depth of melt layers in meter
- Converting m to pixels

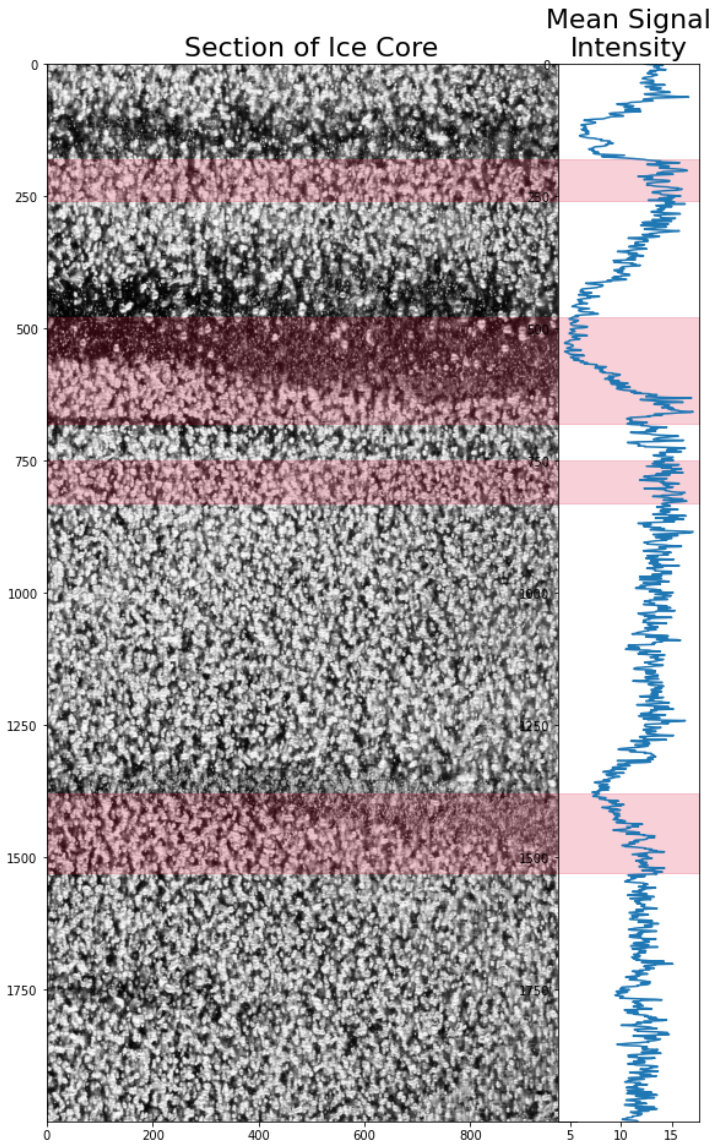
Preprocessing



bag	depth from	depth to	age (b2k)	age (CE)	type of feature
218	120.5650	120.565	858.045	1142.0	meltLens
NaN	121.7200	NaN	868.236	1131.8	meltLayer uncertain
221	122.3063	122.3063	872.875	1127.1	meltLens
NaN	122.3900	NaN	873.636	1126.4	meltLens uncertain
NaN	123.5600	NaN	885.927	1114.1	indication of bubble free layer or patch
224	124.0487	124.0487	891.259	1108.7	meltLens
224	124.2134	124.2134	893.055	1106.9	meltLens
227	124.7216	124.7216	897.066	1102.9	meltLens
227	125.7708	125.7745	905.697	1094.3	meltLayer
230	127.2374	127.2374	917.704	1082.3	meltLens
233	128.1094	128.1094	925.631	1074.4	meltLens

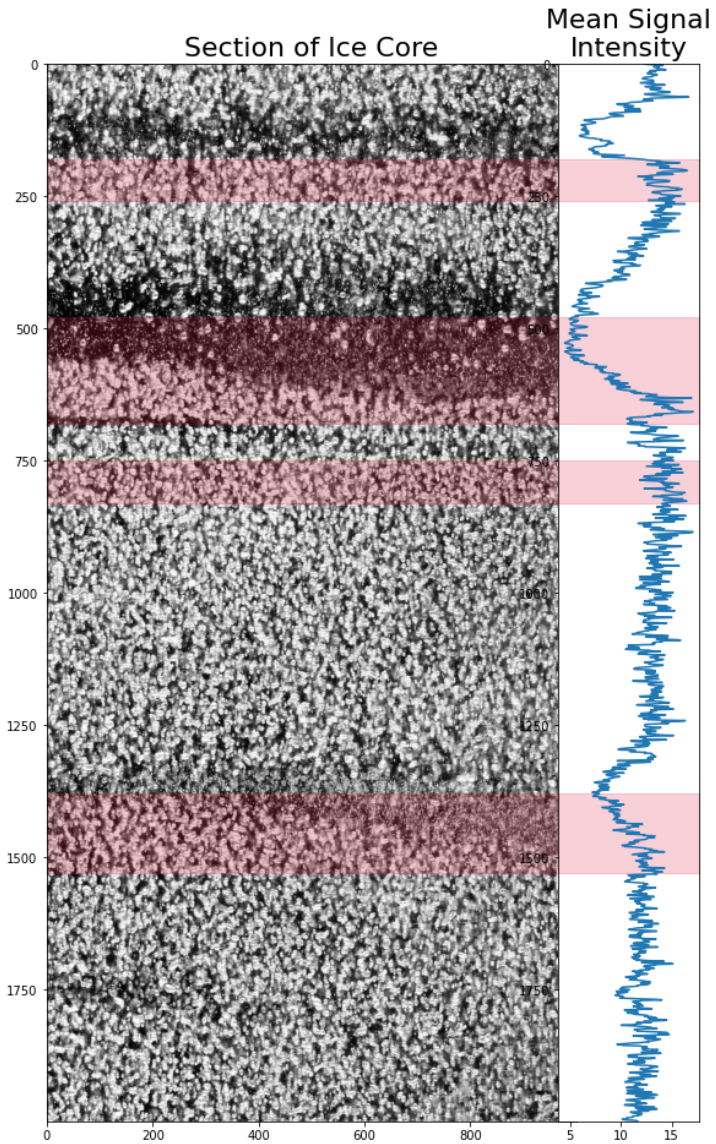
- Table with depth of melt layers in meter
- Converting m to pixels
- Labels don't align with image
- For most images: same offset for all layers
- But not same offset from image to image

Preprocessing II



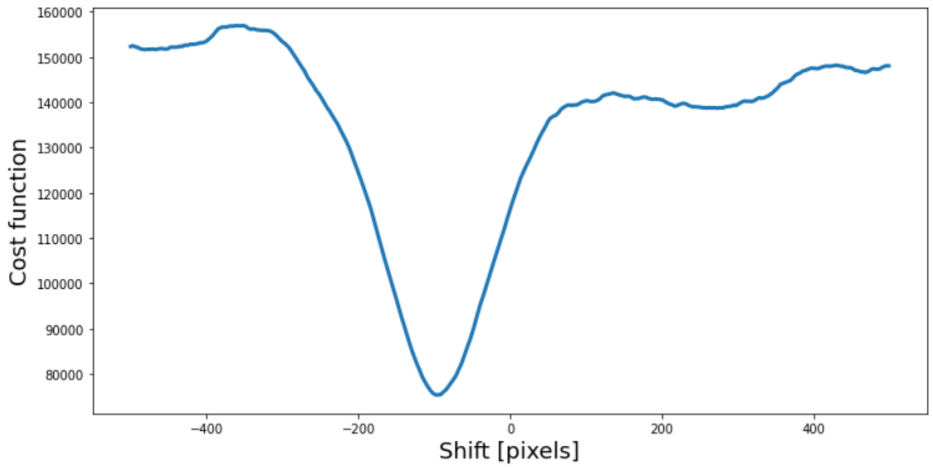
How to find shift value for each image?

Preprocessing II



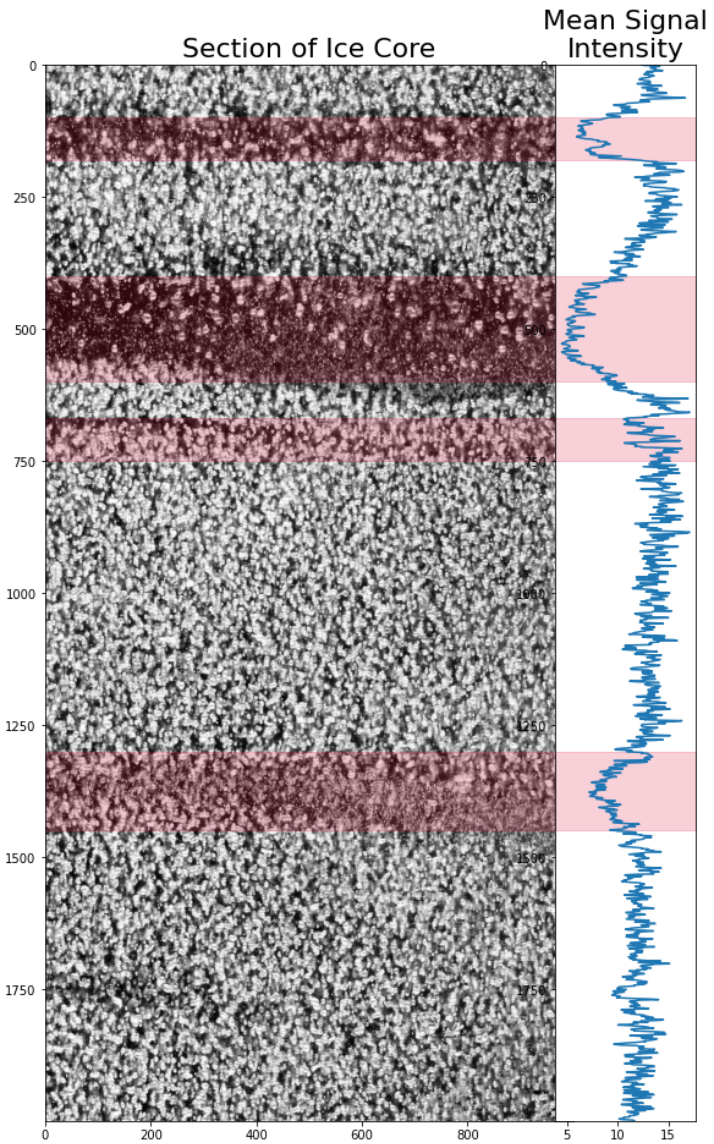
How to find shift value for each image?

$$cost = \sum^{\text{Layers}} mean\,(\text{Signal Intensity [Layer]})$$



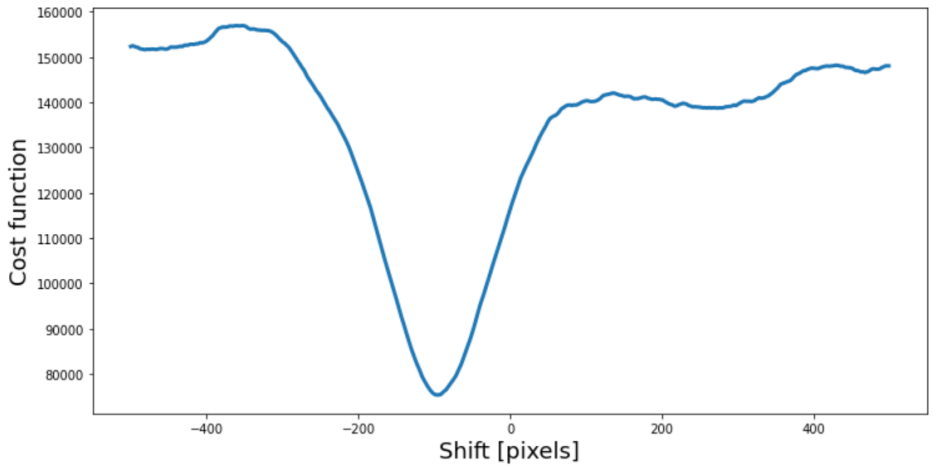
- Try different shift values and choose the one that gives the lowest cost

Preprocessing II



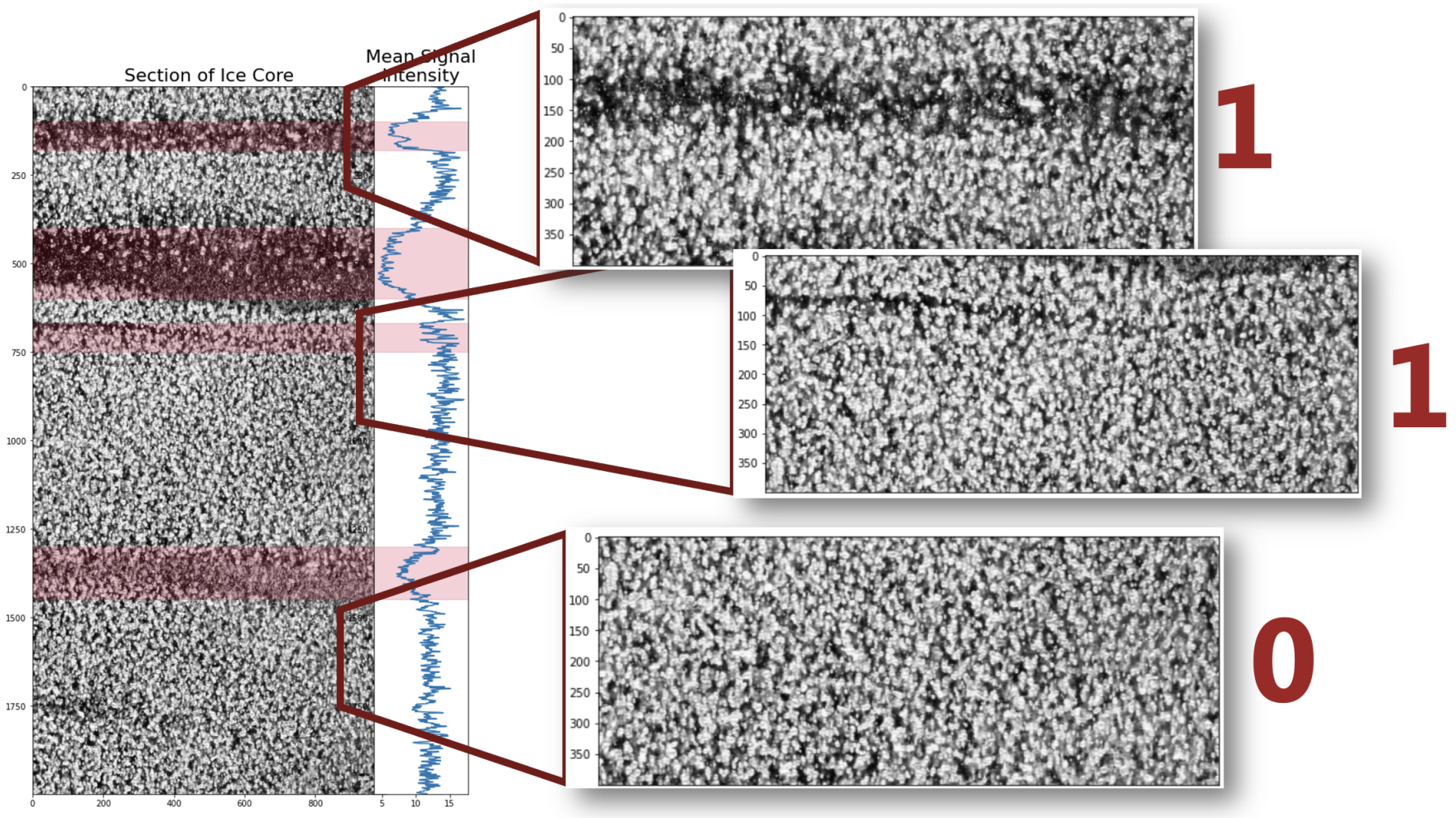
How to find shift value for each image?

$$cost = \sum_{\text{Layers}} mean\,(\text{Signal Intensity [Layer]})$$



- Try different shift values and choose the one that gives the lowest cost

Preprocessing III



This way we generate up to 100 smaller images from one photo

Model Implementations

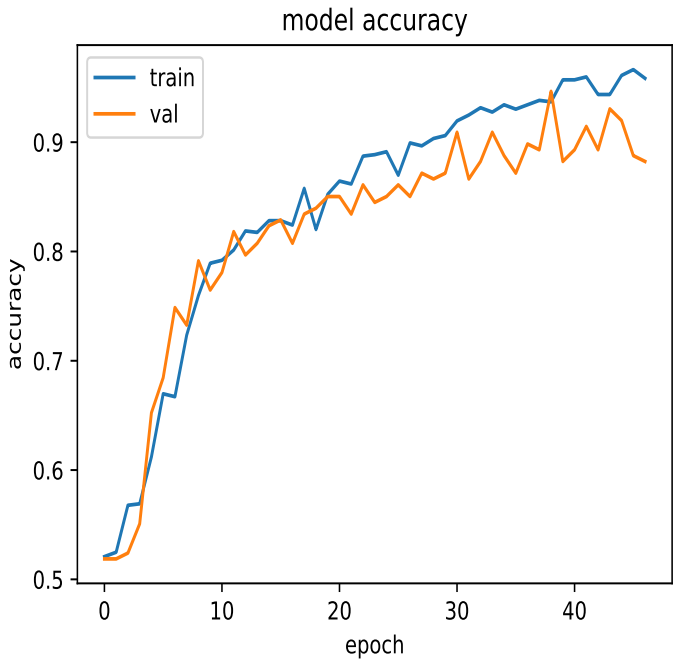
- Simple CNN : Accuracy = 98.6%
- Simple CNN 2.0 : Accuracy = 98.0% (overfitting)
- Enhanced CNN : Accuracy = 97.6 %
- ResNet : Accuracy = 77.9 %
- Pretrained Model: Accuracy = 52.7% (doesn't classify data)

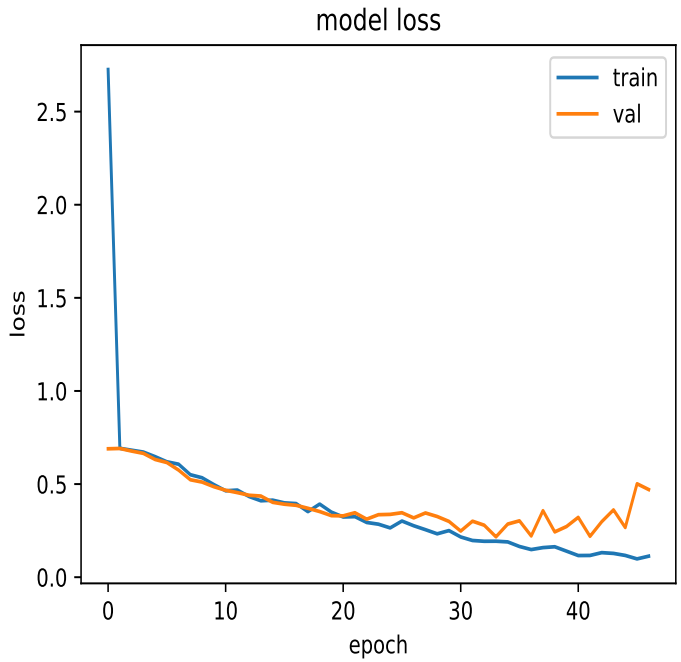
Simple CNN

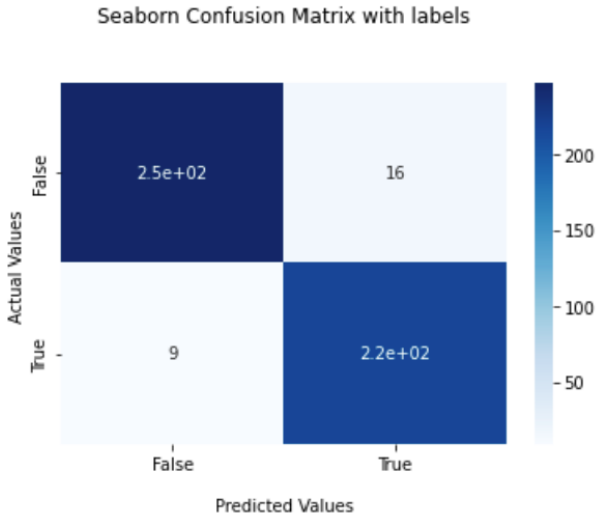
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 50, 200, 32)       896

 max_pooling2d_6 (MaxPooling  (None, 25, 100, 32)      0
 2D)

 dropout_4 (Dropout)         (None, 25, 100, 32)       0

 conv2d_7 (Conv2D)           (None, 25, 100, 64)       18496

 max_pooling2d_7 (MaxPooling  (None, 12, 50, 64)       0
 2D)

 dropout_5 (Dropout)         (None, 12, 50, 64)        0

 conv2d_8 (Conv2D)           (None, 12, 50, 128)       73856

 max_pooling2d_8 (MaxPooling  (None, 6, 25, 128)       0
 2D)

 dropout_6 (Dropout)         (None, 6, 25, 128)        0

 flatten_2 (Flatten)         (None, 19200)             0

 dense_6 (Dense)             (None, 128)               2457728

 dropout_7 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 1)                 129

=================================================================
Total params: 2,551,105
Trainable params: 2,551,105
Non-trainable params: 0
_________________________________________________________________
```

- 200x50 Images
- 3 Convolutional Blocks
- 2 Dense Layers
- Dropout to Avoid Overfitting

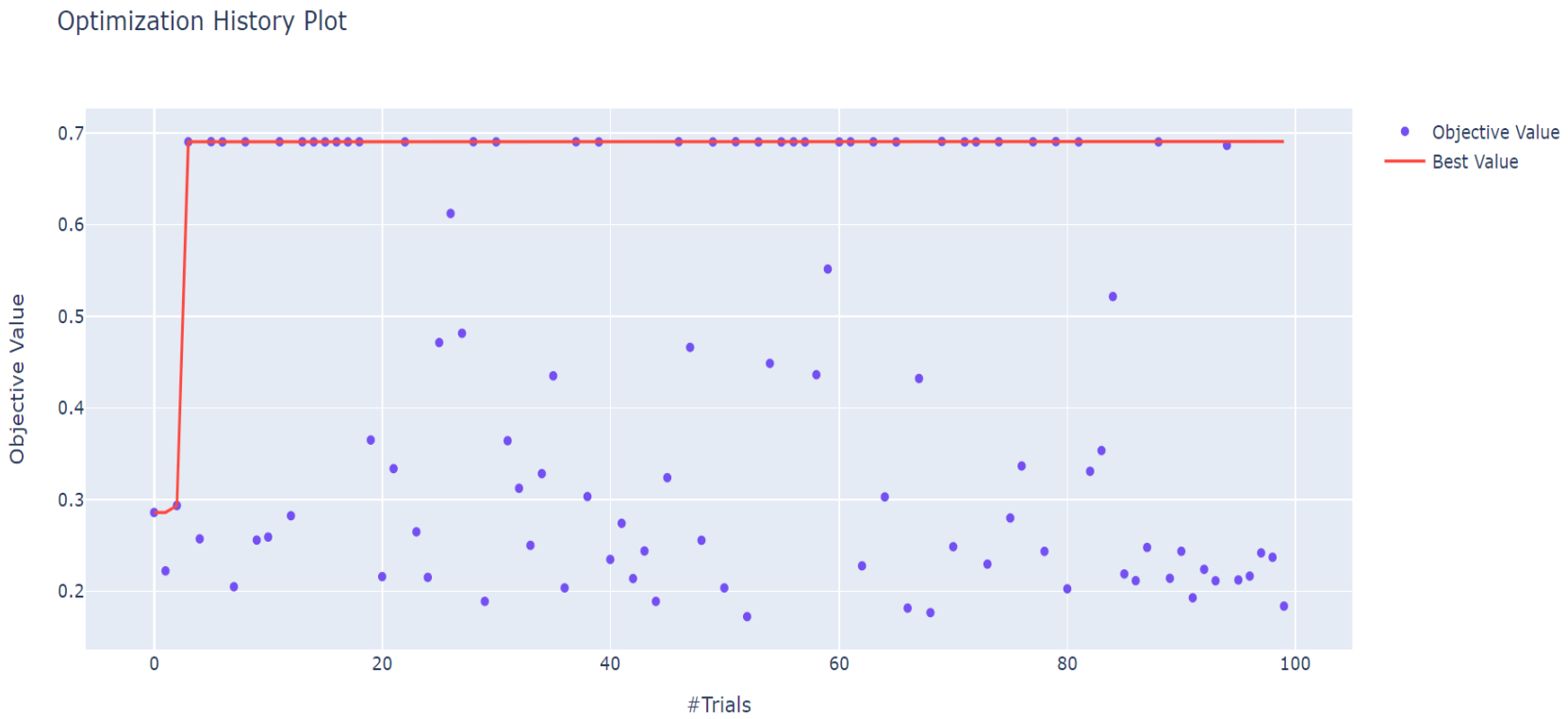
Simple CNN


model accuracy
train
val
accuracy
epoch
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40


model loss
train
val
loss
epoch
0.0
0.5
1.0
1.5
2.0
2.5
0
10
20
30
40

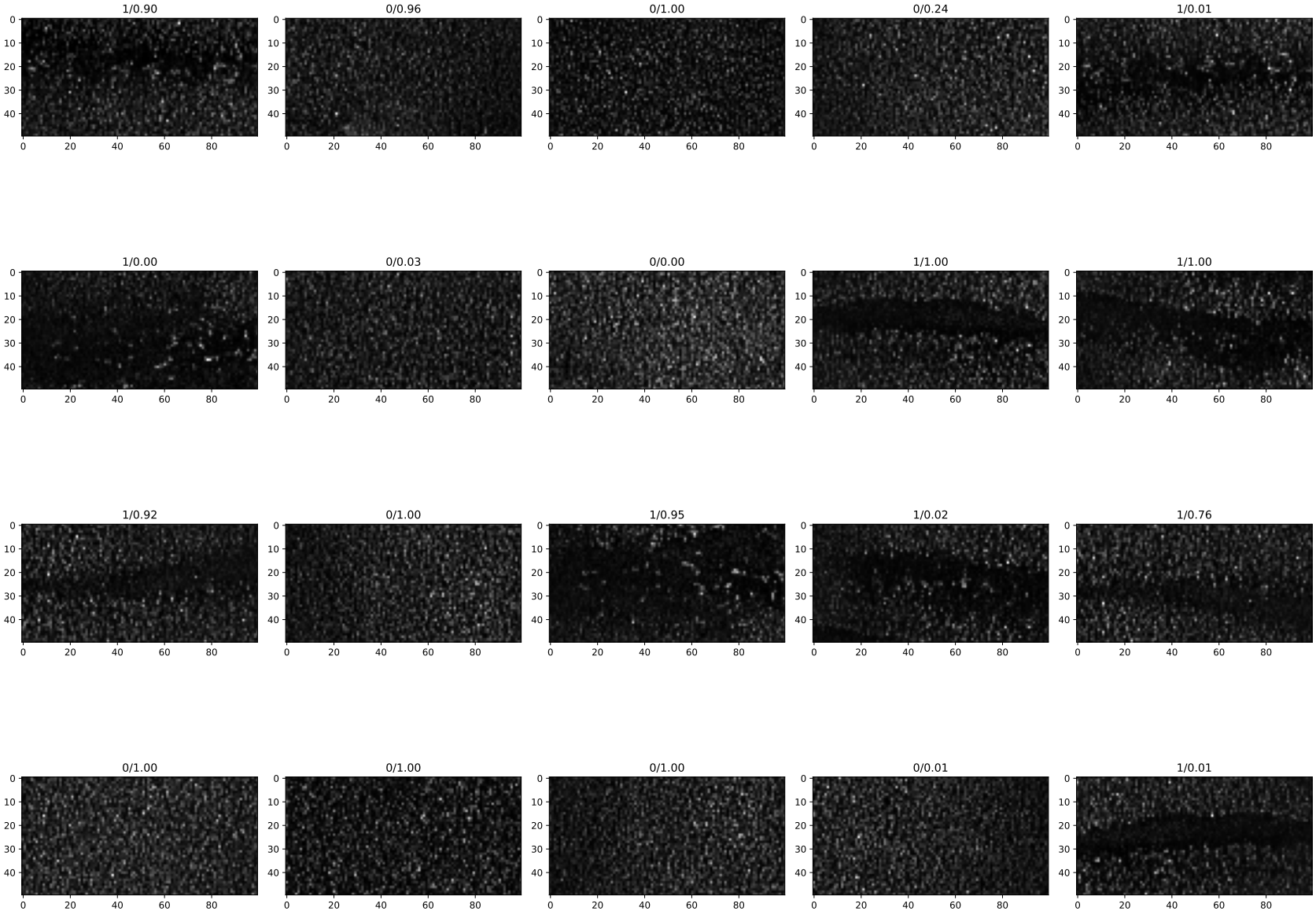

Seaborn Confusion Matrix with labels
2.5e+02
16
9
2.2e+02
False
True
Actual Values
False
True
Predicted Values
200
150
100
50

HyperParameter Optimization

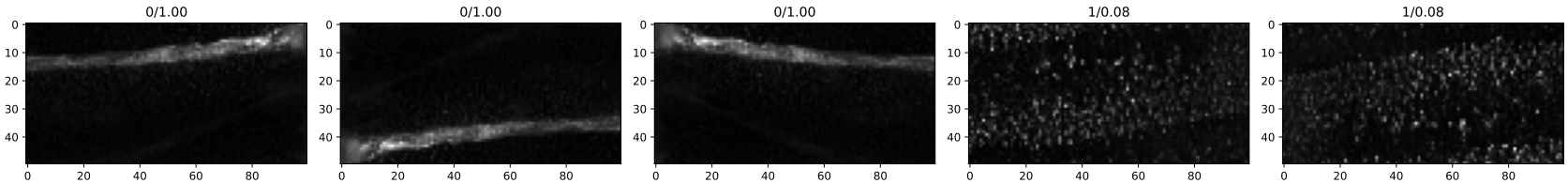


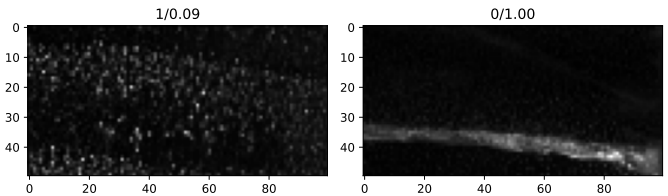
- OPTUNA for Simple CNN, DropOut Rate Optimization

Some Results of the Simple CNN

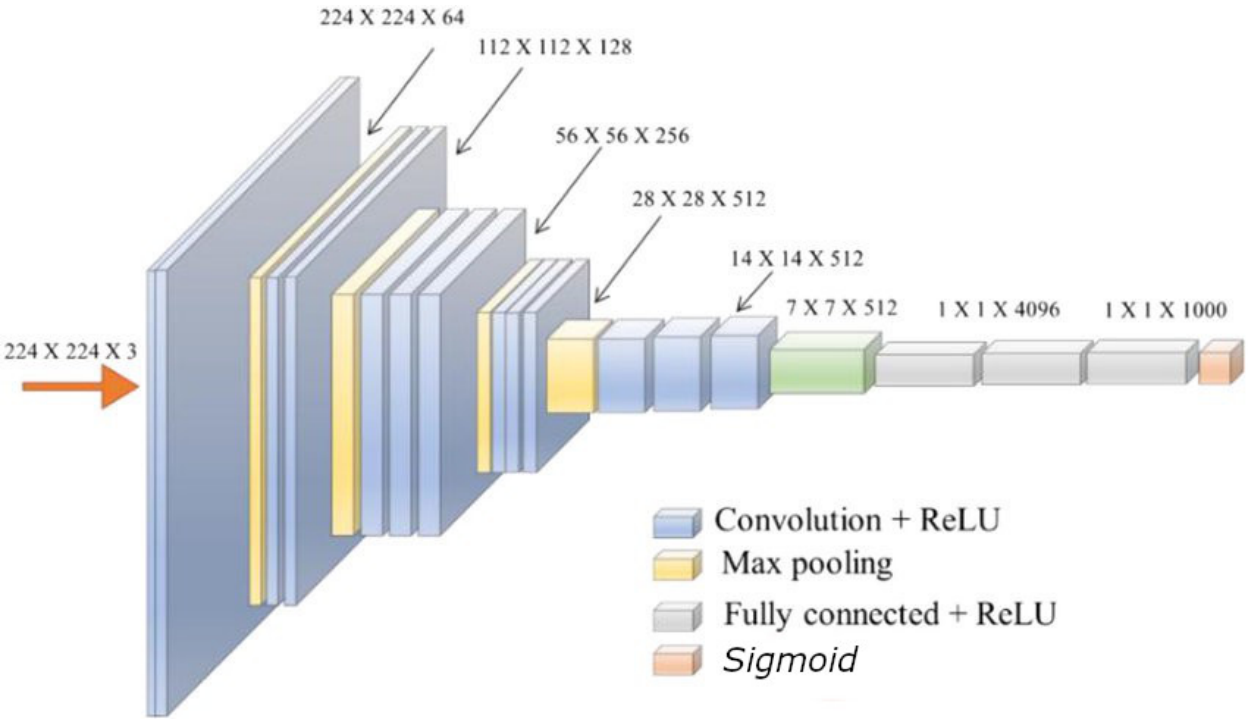


Some Wrong Results of the Simple CNN



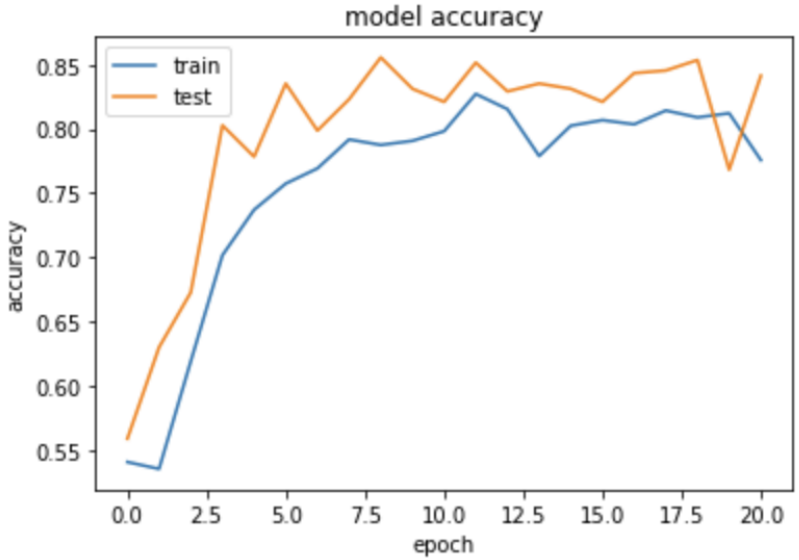


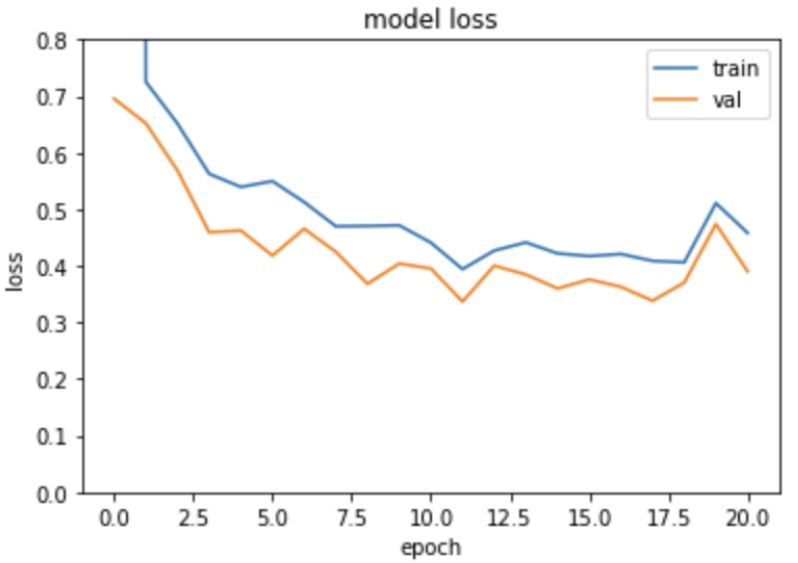
CNN with Tranfer Learning

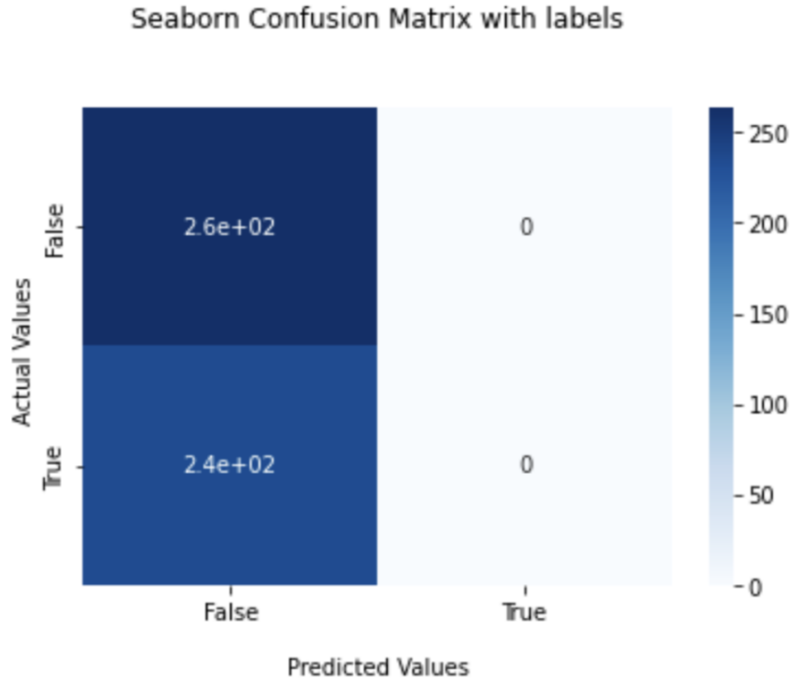


- VGG-16
  - Pre-trained Layers Frozen center
  - 3 Dense "Top" Layerscen-ter
  - 224x224 Images

CNN with Tranfer Learning



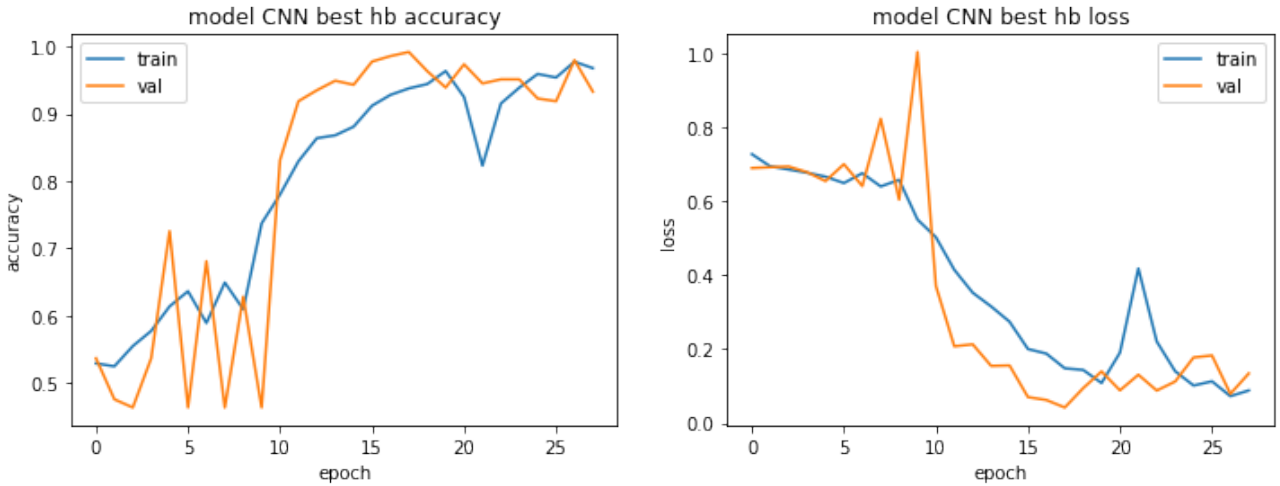


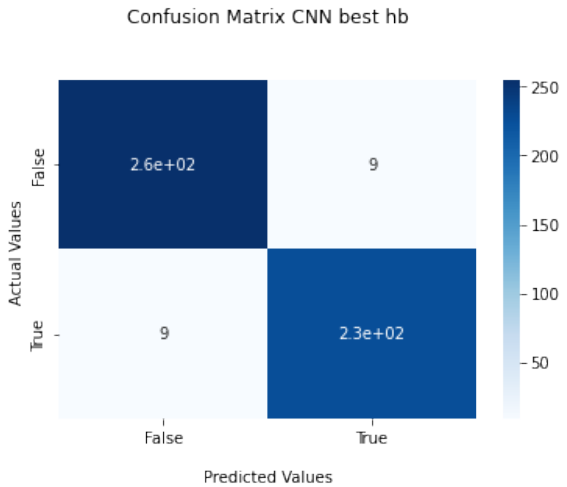


Enhanced CNN

- CNN with convolution layers pooled and max pooling
- Optimization with the Hyperband algorithm (Cite) via Keras-tuner
  - It samples the hyper parameter space with a limited number of epochs initially to learn about the space then iterates through more epochs for the more promising models.
  - variables:
    - input shape (50, 200, 3)
    - Optimizer (sgd: momentum=0.9, nesterov=True ; nadam: beta1=0.9, beta2=0.999)
    - Learning rate: from 10-2 to 10-4
    - Number of filters to start: 64 to 256
    - Number of convolutional layers in sequence: 2 to 4

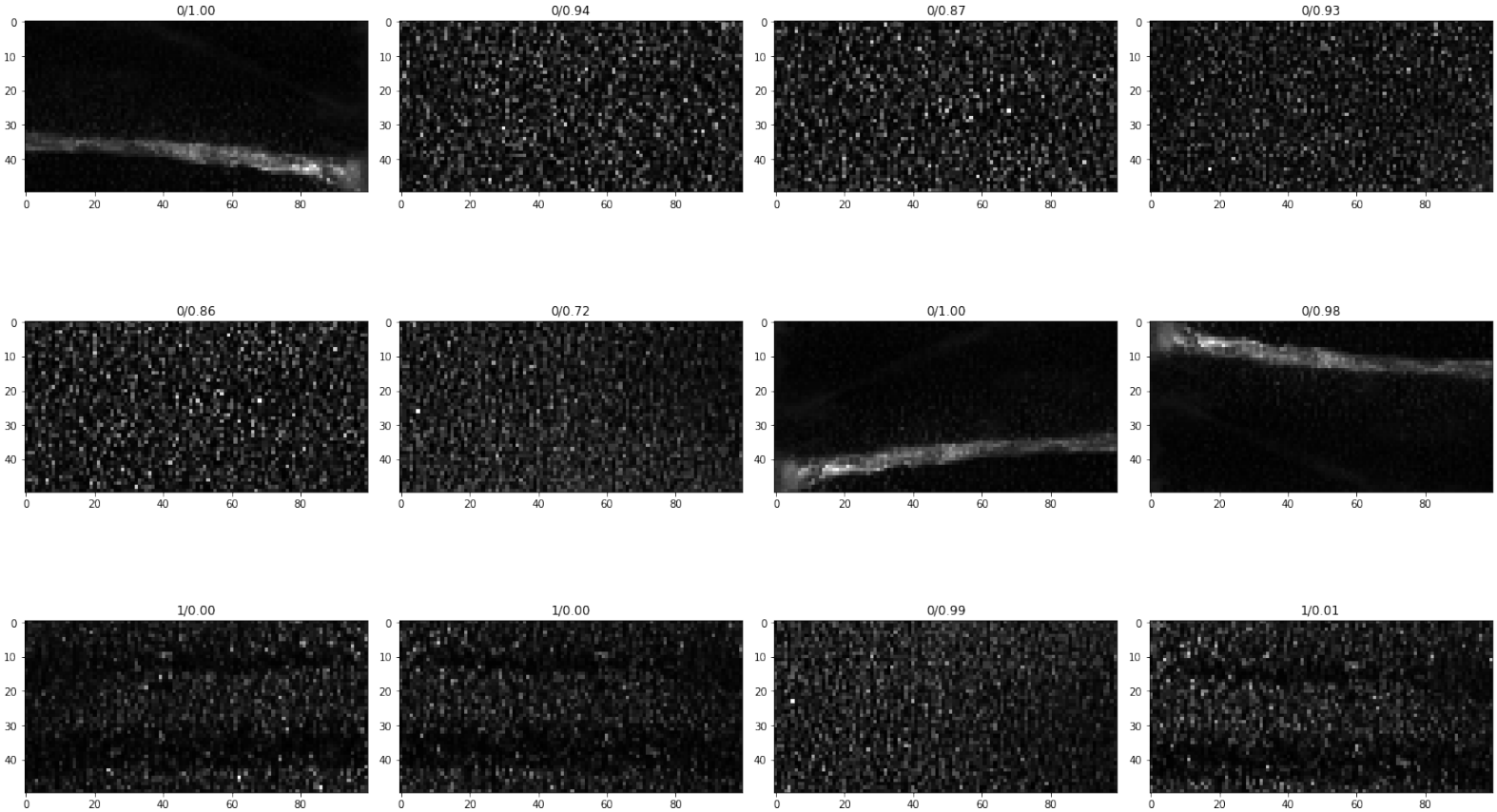
Enhanced CNN: Results



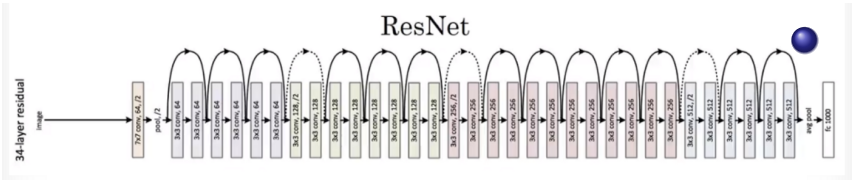


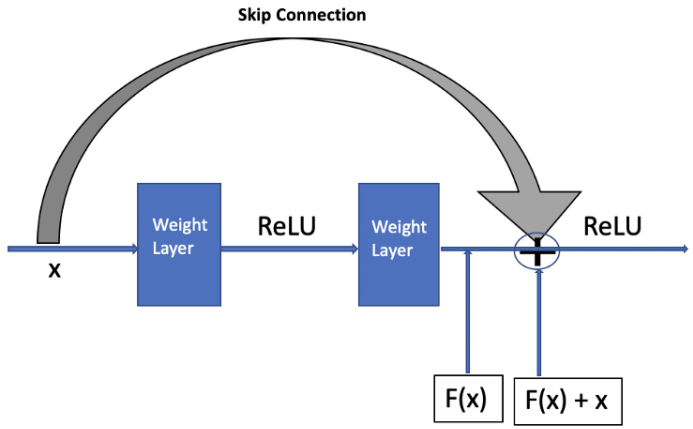
- Results: Time used on GPU: 74793.5 ms; Test loss: 0.121 / Test accuracy: 0.9760

Enhanced CNN: wrong results



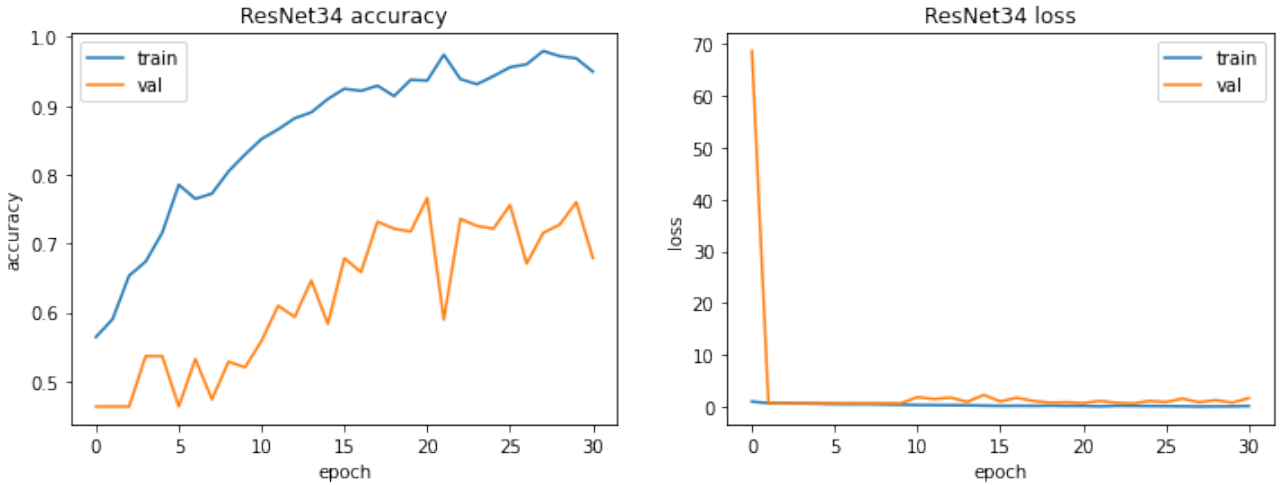
ResNet

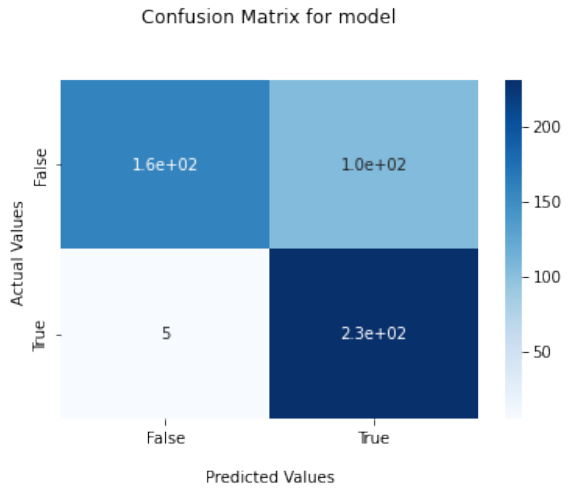




- Residual Neural Network
- Residual learning via Residual Units and skip connections
  - $H(x) = F(x) + x$
  - speed up training considerably: it initially models the identity function. If the target function is close to the identity function the training is speed up
  - the network will make progress even if the several layers have not started learning yet

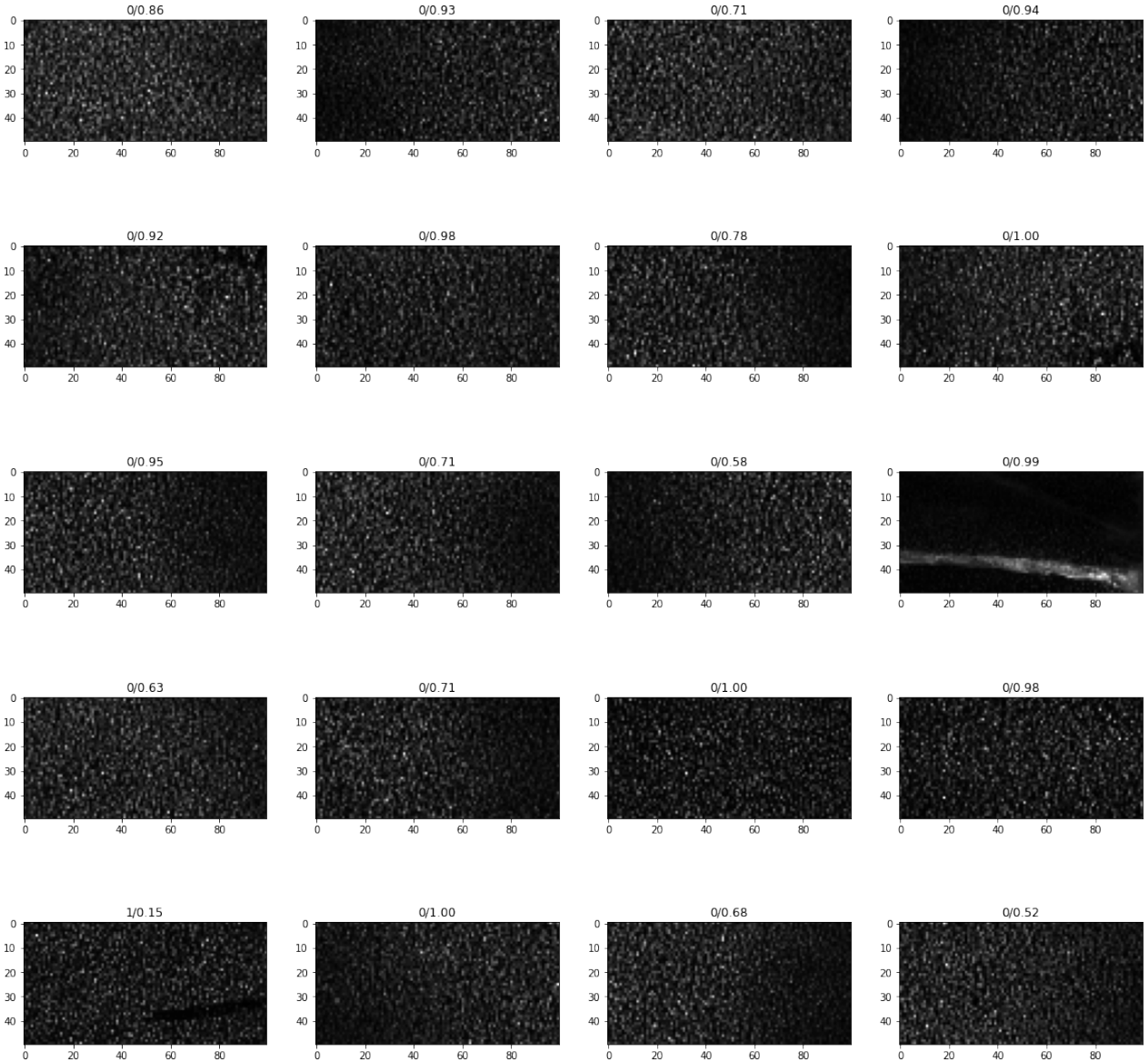
Enhanced CNN: Results





- Results: Time used by ResNet34 on GPU: 132299.0 ms ; Test loss: 0.67 / Test accuracy: 0.779

ResNet34: Wrong results



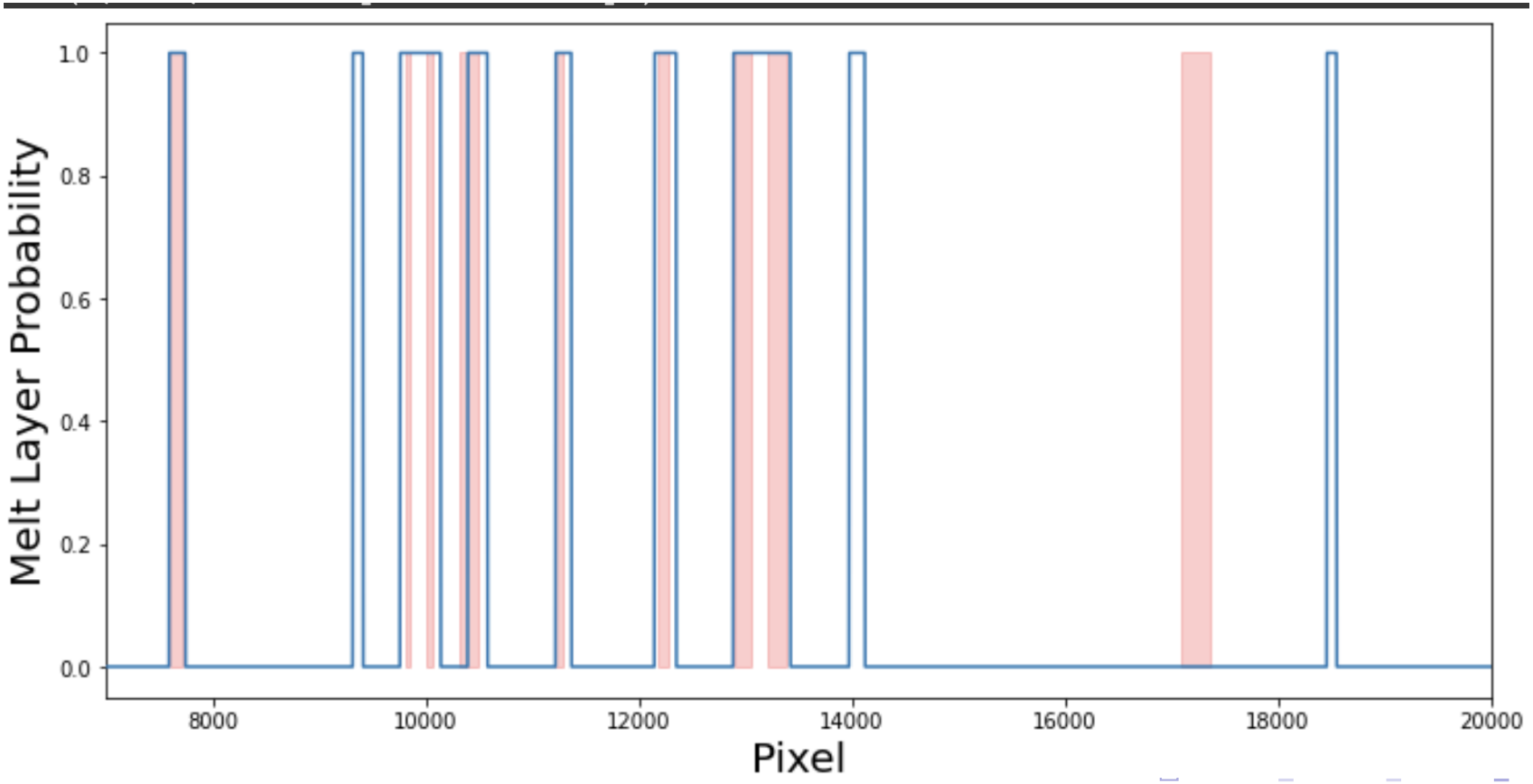
Enhanced CNN

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 50, 200, 157)      11932

 max_pooling2d (MaxPooling2D  (None, 25, 100, 157)     0
 )

 conv2d_1 (Conv2D)           (None, 25, 100, 157)      221998

 conv2d_2 (Conv2D)           (None, 25, 100, 157)      221998

 max_pooling2d_1 (MaxPooling  (None, 12, 50, 157)      0
 2D)

 conv2d_3 (Conv2D)           (None, 12, 50, 314)       443996

 conv2d_4 (Conv2D)           (None, 12, 50, 314)       887678

 max_pooling2d_2 (MaxPooling  (None, 6, 25, 314)       0
 2D)

 conv2d_5 (Conv2D)           (None, 6, 25, 471)        1331517

 conv2d_6 (Conv2D)           (None, 6, 25, 471)        1997040

 max_pooling2d_3 (MaxPooling  (None, 3, 12, 471)       0
 2D)

 conv2d_7 (Conv2D)           (None, 3, 12, 628)        2662720

 conv2d_8 (Conv2D)           (None, 3, 12, 628)        3550084

 max_pooling2d_4 (MaxPooling  (None, 1, 6, 628)        0
 2D)

 flatten (Flatten)           (None, 3768)              0

 dense (Dense)               (None, 314)               1183466

 dropout (Dropout)           (None, 314)               0

 dense_1 (Dense)             (None, 157)               49455

 dropout_1 (Dropout)         (None, 157)               0

 dense_2 (Dense)             (None, 1)                 158

=================================================================
Total params: 12,562,042
Trainable params: 12,562,042
Non-trainable params: 0
_________________________________________________________________
```

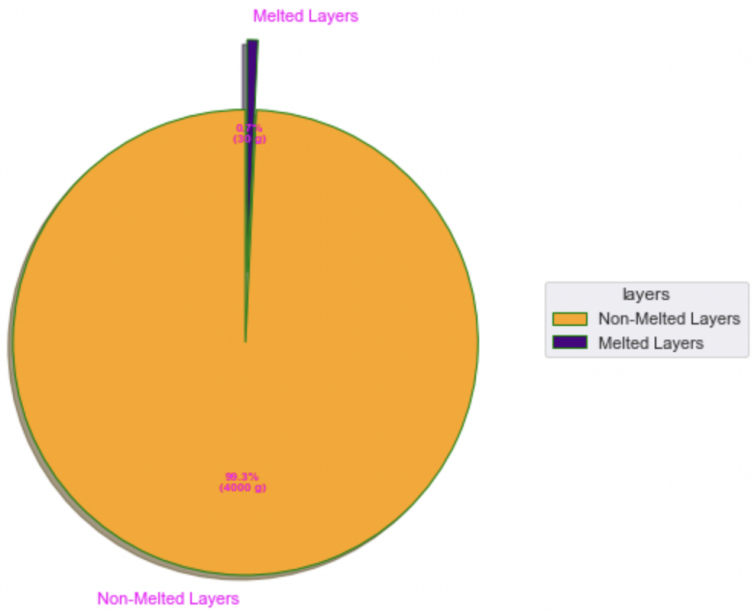
- 200x50 Images
- 4 Convolutional Blocks with max pooling in between
- 3 Dense Layers incl. Dropout to Avoid Overfitting

Using the Model

- Apply preprocessing to image (cutting sides, top and bottom)
- Slice image in multiple slices (stride 10 pixels)
- Flip (horizontal and or vertical)
- Predict on each (flipped) slice –> 4 predictions
- Most common prediction



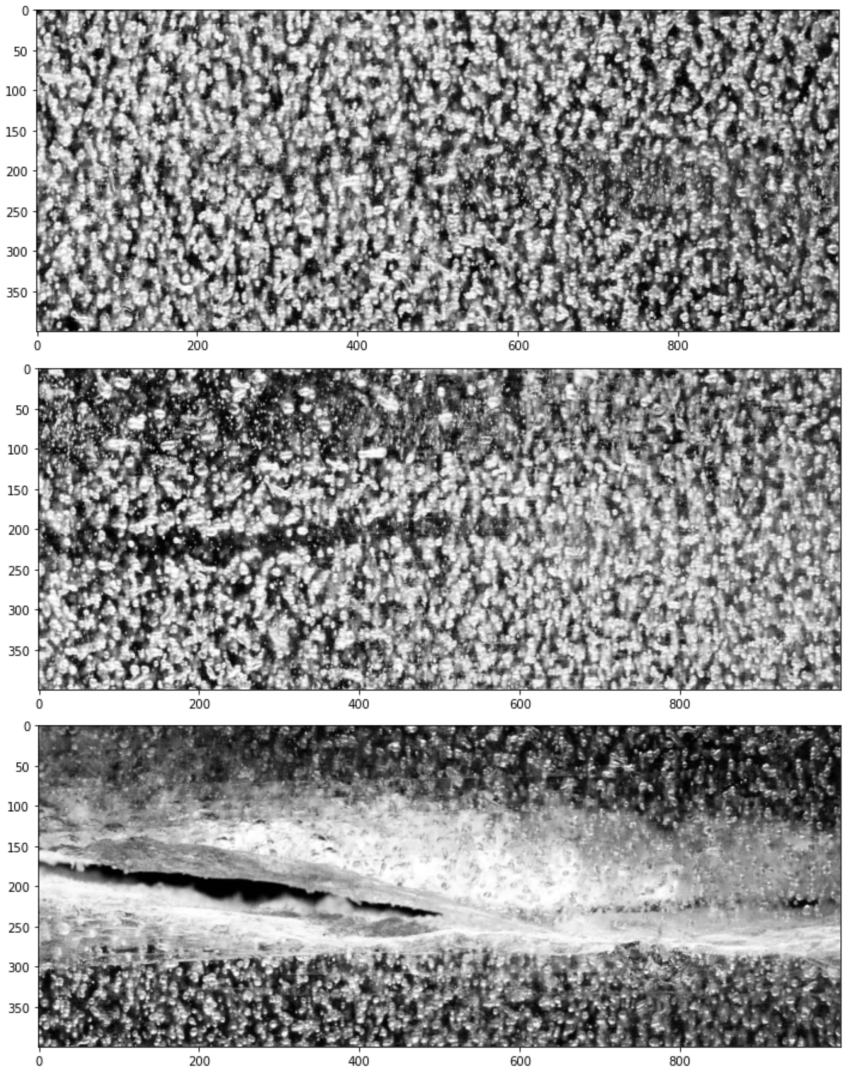
Where did it go wrong?



Unbalance Dataset

- Data Augmentation is not enough

Where did it go wrong?

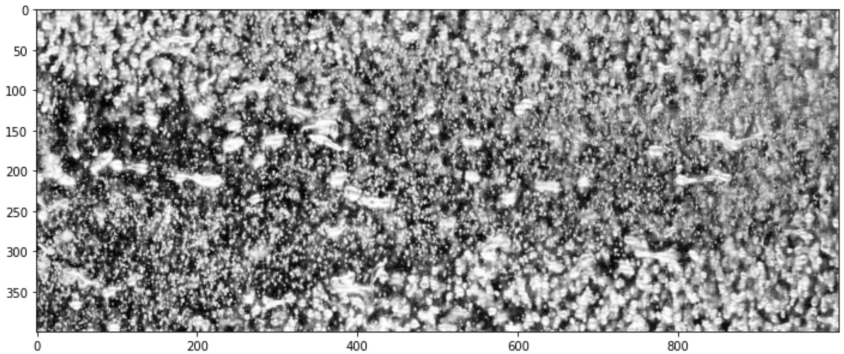


False Positives

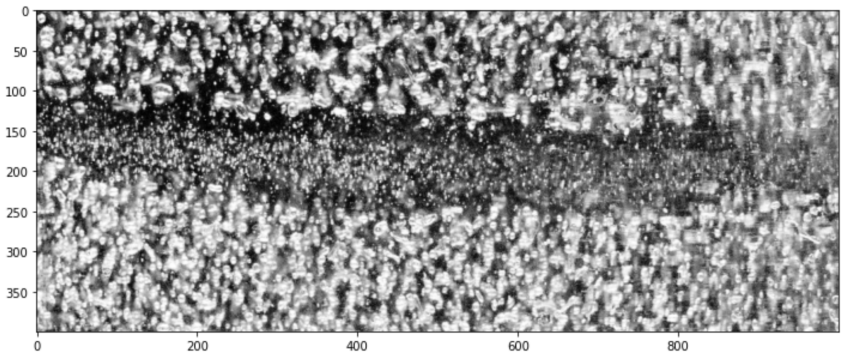
- There is a chance that melt lenses are predicted as layers
- Cracks in ice core are almost always predict to be a layer

Where did it go wrong?

False Negative example

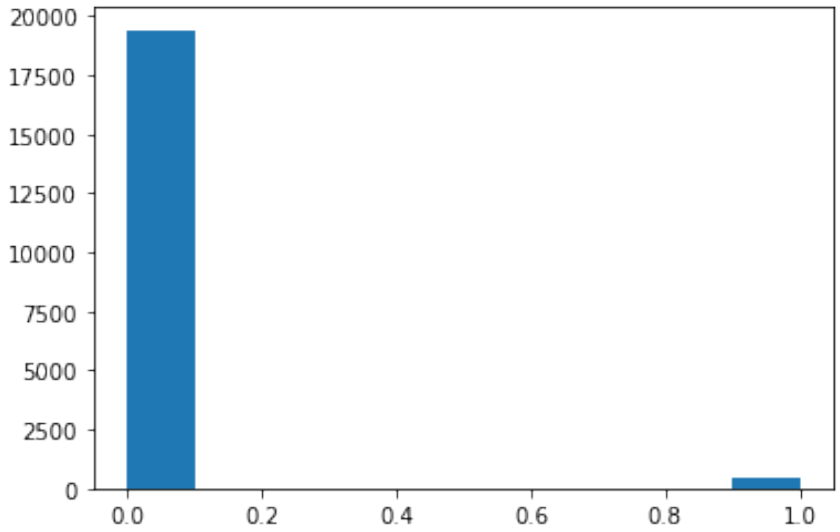


compared to True Positive



- Diffuse melt layer imo

Possible way to fix the problem




```
Epoch 11/50
901/901 [==============================] - 735s 815ms/step - loss: 0.1089 - accuracy: 0.9522 - val_loss: 0.1080 - val_accuracy: 0.9634
```

- Generate an imbalanced training set with a lot of non-melt layers for the model to learn e.g. cracks, etc.
  - After Combining:
  - Training Set: 19819 with 428 1s and 19391 0s
  - Validation Set: 492 with 228 1s and 264 0s
  - Test Set: 500 with 236 1s and 264 0s
  - currently running ResNet34 model on CPU (11/50 epochs in 142 min)

Conclusion

- Tuned Models with low False Negative could be used to select portion of image to then manually label (as melt layer / melt lens)
- Larger Dataset is needed
- **Alternative approaches**
  - Object Localization with CNN
    - Last layer doesn't predict one float (probability) but instead two integers (start and stop of melt layer in pixels)
    - Would then be a regression task

1 Appendix

Appendix A1

General setup: Training Set: 932 with 428 1s and 504 0s Test Set: 500 with 236 1s and 264 0s Validation Set: 492 with 228 1s and 264 0s

Appendix A2

- Optimization with the Hyperband algorithm. Top 3 models:
- Best model: 'conv layers': 4, 'filters': 157, 'learning rate': 0.004940653067215263 4, 'optimizer': 'sgd'
- 2nd best model: 'conv layers': 3, 'filters': 251, 'learning rate': 0.0001377968661129521, 'optimizer': 'nadam'
- 3rd best model: 'conv layers': 3, 'filters': 232, 'learning rate': 0.0002284305335720264, 'optimizer': 'nadam'

Simple CNN 2.00

Image pre-processing: Cutting edges and Image sclicing in 30 images of 1000x1000 pixels

Data augmentation:

keras.preprocessing.image.ImageDataGenerator(brightness_range, horizontal_flip, vertical_flip)

Data set:

30 images with MeltLayer/MeltLayers

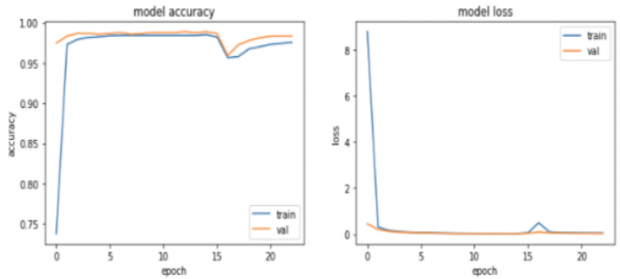
35 images with no MeltLayer

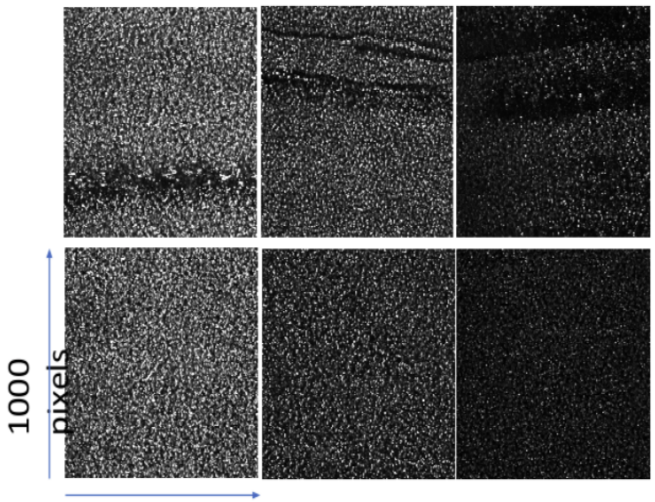
Final data set: 2976 images with no MeltLayer

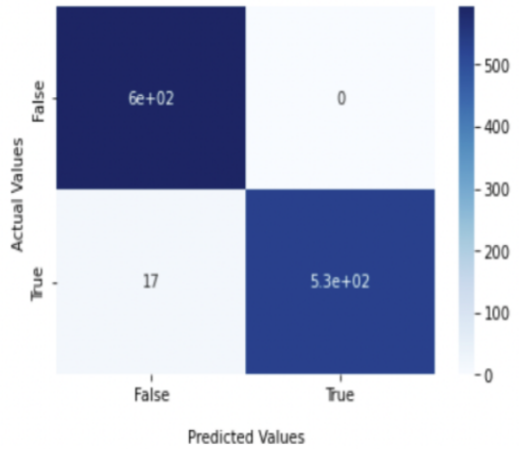
2734 images with MeltLayer/MeltLayers

Different CNN architectures did not work

- Input shape (1000, 1000, 3) or (500, 500, 3) - Image downsize
- Validation split: 0.2







Pre-trained model + CNN architecture for inpute shape (1000, 1000, 3)

Need for computation power