Identifying Insolubles in Ice Core Data

Applied Machine Learning Final Project

Lars Erik J. Skjegstad (zfj803)
Martin Lyskjær Frølund (rhk101)
Liam Ward (fqg904)

UNIVERSITY OF COPENHAGEN

Introduction to the problem and the dataset

The datasets

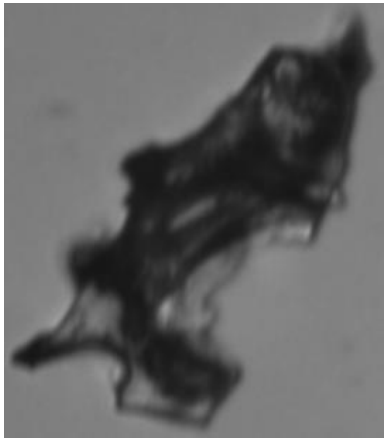
- Real
  - Greenland ice core data (GRIP)
  - 4 ice cores
  - 14-17,000 years old
  - Supplied by Niccolò Maffezzoli
  - 3,085,063 images
  - Not labelled

- Synthetic
  - 136,022 images
  - 6 classes

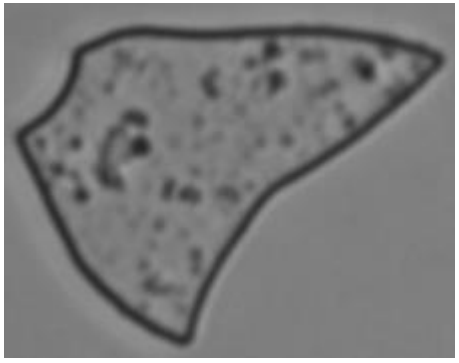
Classes

Ash (55.2%)

Grimsvotn (10.8%)

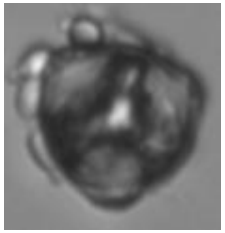


Campanian (44.4%)

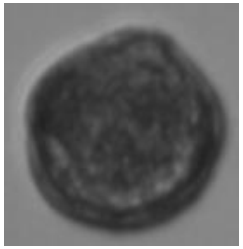


Pollen (22.3%)

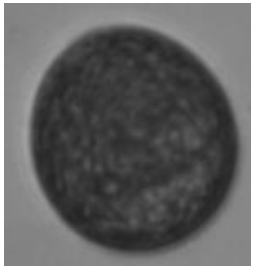
Corylus (5.8%)



Qrobur (8.1%)



Qsuber (8.4%)



Dust (22.6%)

Introduction to the problem

- Classification:
  1. Three-class classification
  2. Six-class classification
  3. Classification of the three pollen types
- Utilizing both meta-data and images
- Real data:
  - Separate "interesting" particle types from dust ("background")

Methods

- CNNs with dense network classifier
  - Keras
  - PyTorch
- Classification using metadata and images
  - LightGBM
  - Keras NN
- Unsupervised learning: UMAP

CNNs with dense network classifier

Pre-processing, data augmentation, setup | CNNs

- Balancing dataset or not?
- Pre-processing: Resizing (1x128x128) and normalization
- Augmentation:
  - Random Crop, Flip and Gaussian Blur
- Early Stopping

Architectures | CNNs

PyTorch:

A1:
Conv(11x11)
MaxPool(2x2)
Conv(5x5)
MaxPool(2x2)
Flatten
Dense
Dense
|
Output

A2:
Conv(11x11)
MaxPool(2x2)
Conv(5x5)
MaxPool(2x2)
Flatten
Dense
BatchNorm
DropOut
Dense
|
Output

A3:
Conv(11x11)
MaxPool(2x2)
Conv(5x5)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Flatten
Dense
BatchNorm
DropOut
Dense
|
Output

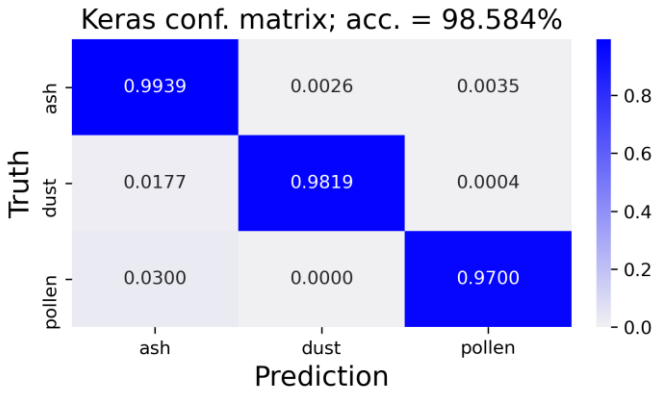
A4:
Conv(11x11)
MaxPool(2x2)
Conv(5x5)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Flatten
Dense
BatchNorm
DropOut
Dense
|
Output

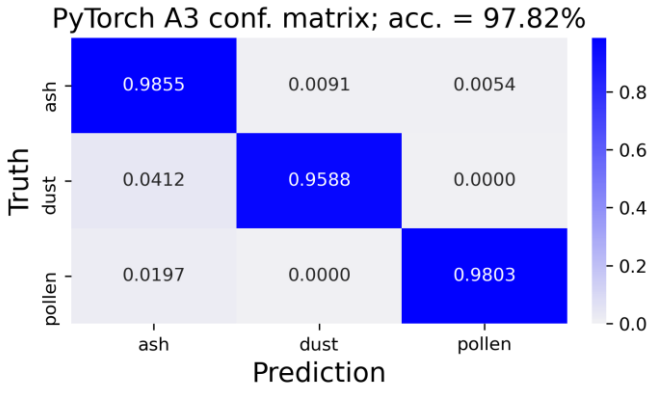
Keras:

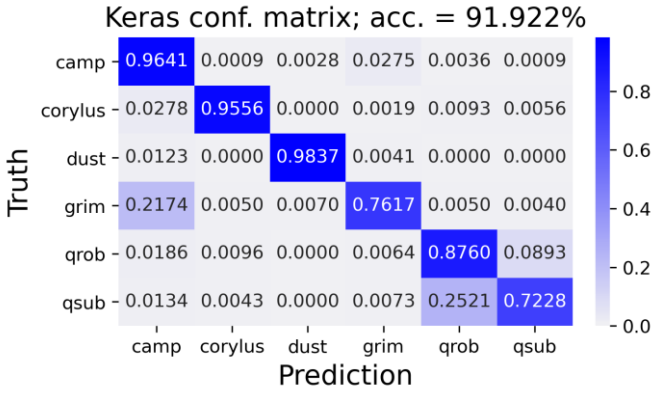
K1:
Conv(3x3)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Conv(3x3)
MaxPool(2x2)
Flatten
Dense
Dense
|
Output

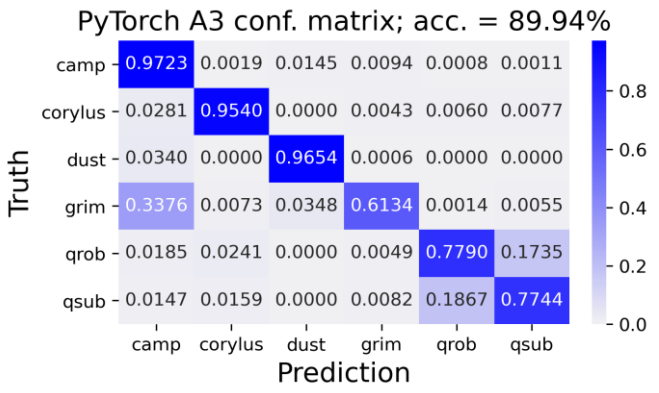
Results | CNNs

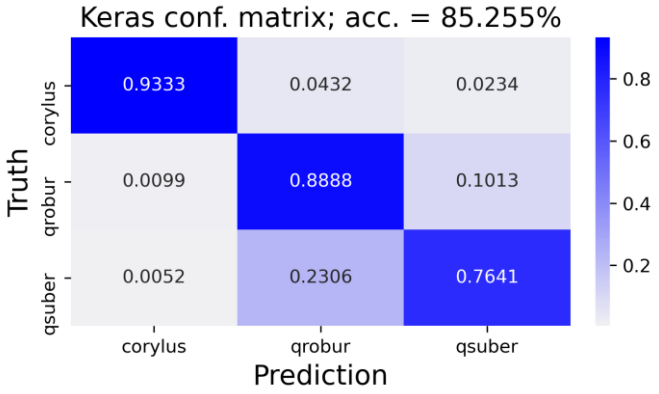
- Three-class classification
- Six-class classification
- Pollen classification

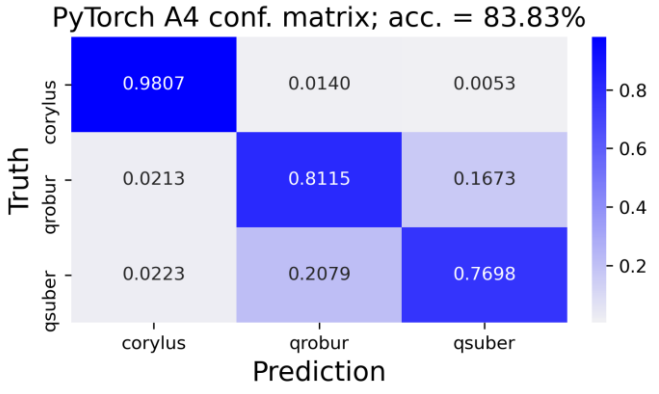










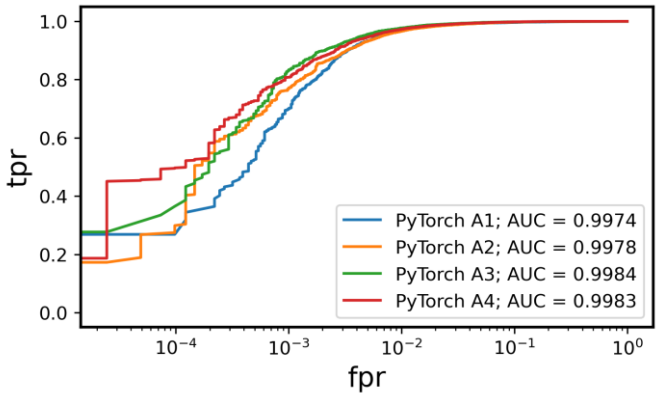


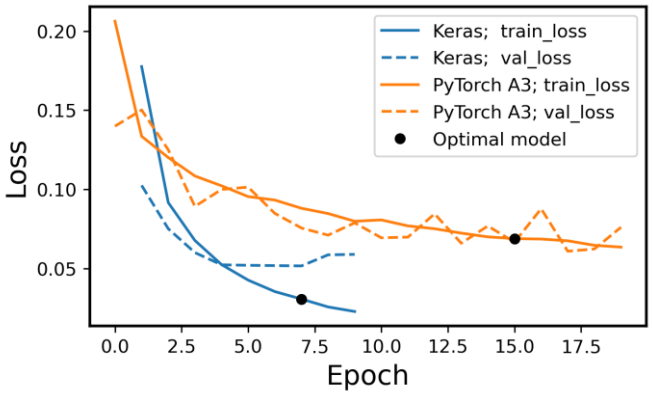
Results | CNNs

ROC curves

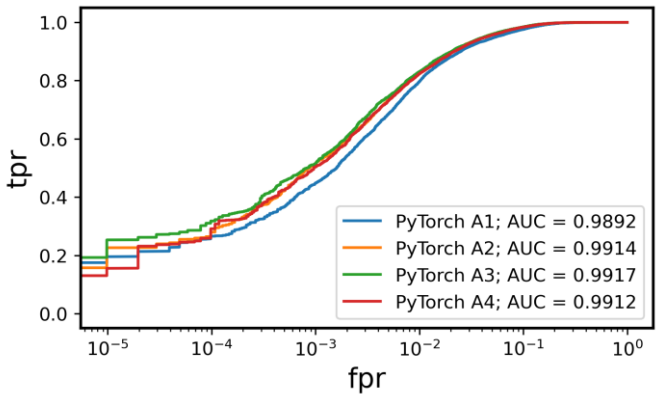
Train and Val. Loss

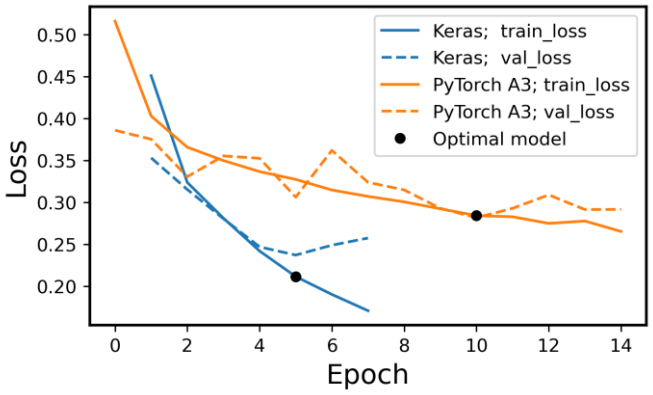
- Three class classification



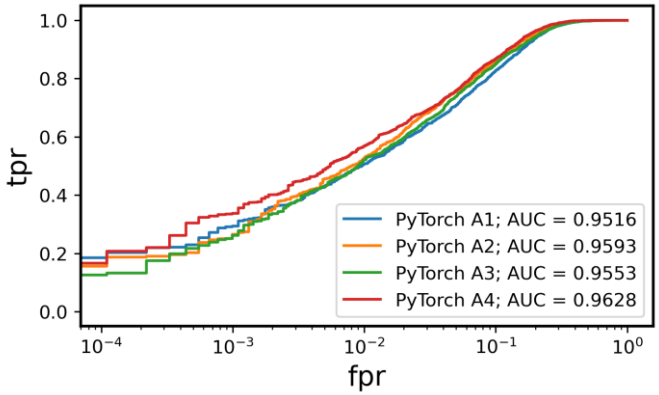


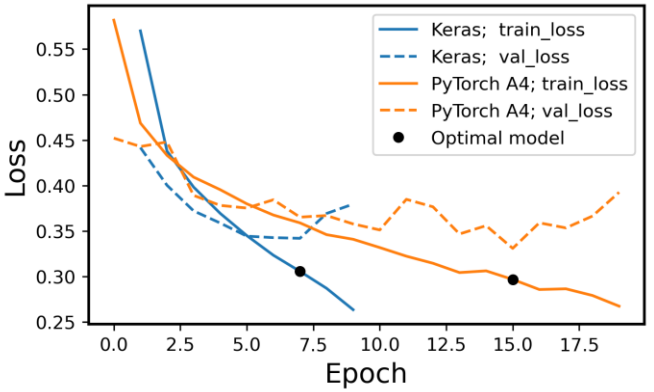
- Six class classification





- Pollen classification





Results | CNNs

- Three class classification

PyTorch A3
Actual consumption for 20 epoch(s):
Time: 0:30:46
Energy: 0.028414 kWh
CO2eq: 8.359634 g
This is equivalent to: 0.069432 km travelled by car

- Six class classification

PyTorch A3
Actual consumption for 14 epoch(s):
Time: 0:24:04
Energy: 0.021607 kWh
CO2eq: 6.357014 g
This is equivalent to: 0.052799 km travelled by car

- Pollen classification

PyTorch A4
Actual consumption for 20 epoch(s):
Time: 0:07:28
Energy: 0.006822 kWh
CO2eq: 2.007057 g
This is equivalent to: 0.016670 km travelled by car

Results | CNNs

- Three-class c

- S c

- Pollen classification

PyTorch A3
Actual consumption for 20 epoch(s):

by car

by car

PyTorch A4
Actual consumption for 20 epoch(s):
Time: 0:07:28
Energy: 0.006822 kWh
CO2eq: 2.007057 g
This is equivalent to: 0.016670 km travelled by car

Approx. total consumption:

8 g/class. • 3 class. • 3 persons
• 30 runs = **2.16 kg**
or **18 km** travelled by car!

Classification using metadata and images

Pre-processing | Metadata and images

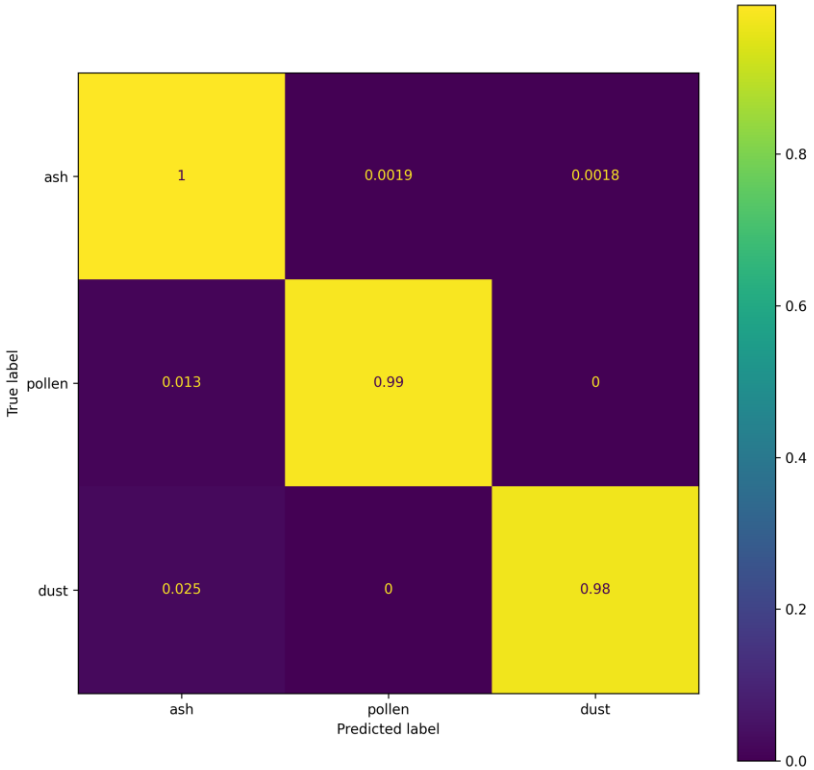
- LightGBM (Tree-based)
  - No pre-processing required
  - Simply include only the relevant features
  - Label according to the classification required

- IsolationForest
  - Train only on the 'normal' class (Dust)
  - Fits a decision boundary that determines the support of what is normal.
  - Predict anomaly scores for unseen data. More negative scores indicate more anomalous. I.e. Interesting!

Classification: MetaData and Transfer Learning LightGBM

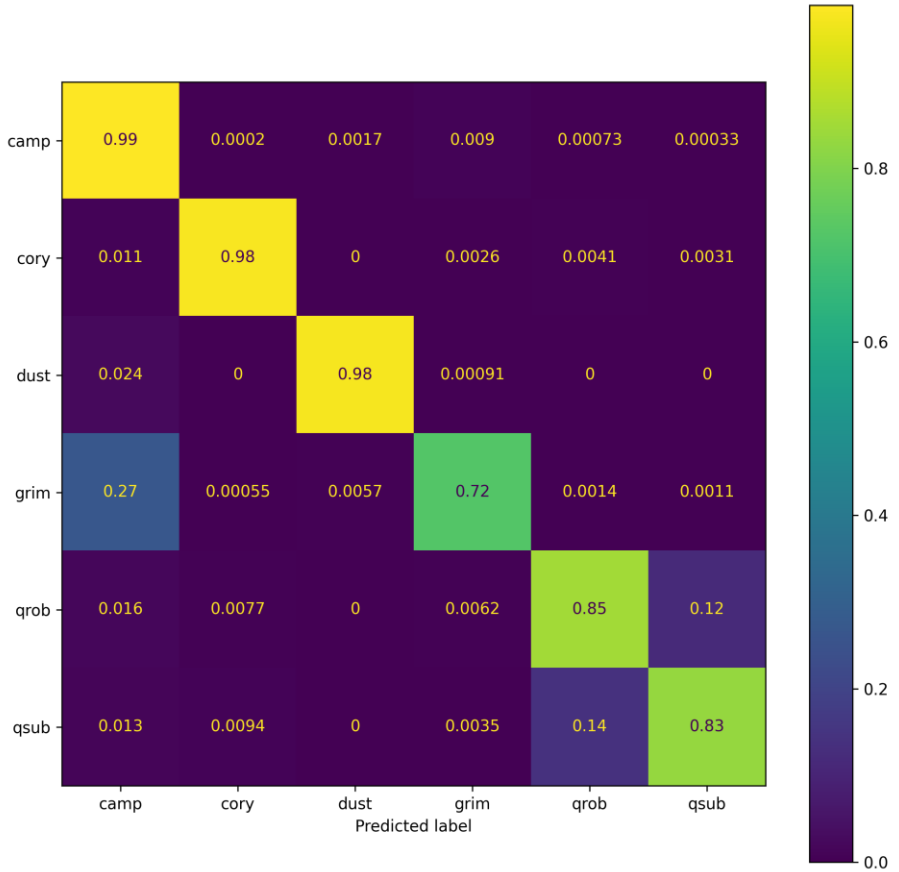
3 class accuracy: 0.99

Confusion Matrix: 3 Class



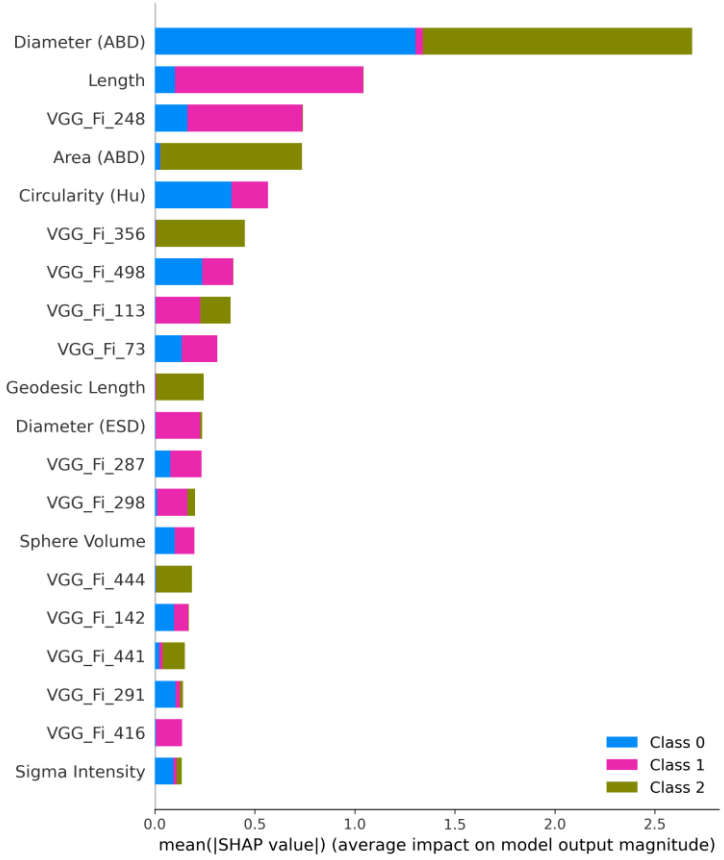
6 class accuracy: 0.92

Confusion Matrix: 6 Class

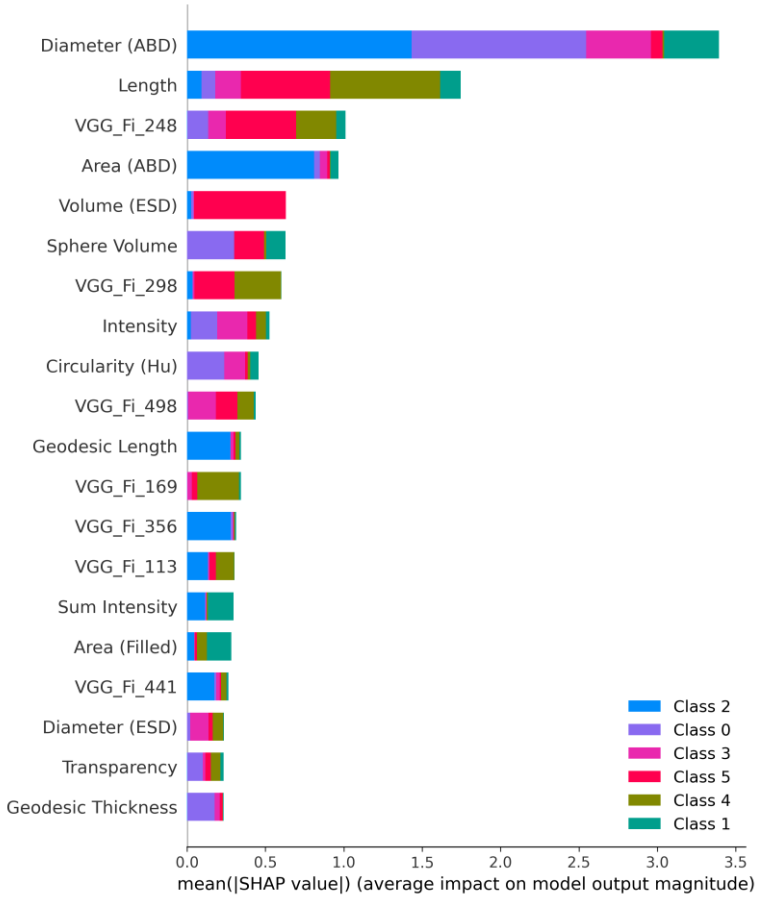


Classification: MetaData and Transfer Learning LightGBM

Shap values: 3 Class

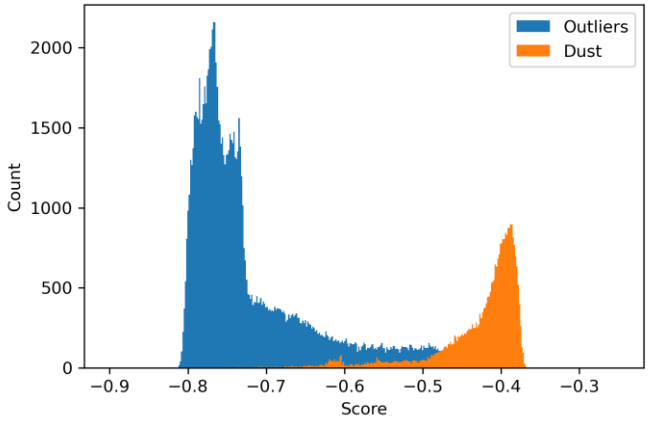


Shap values: 6 Class

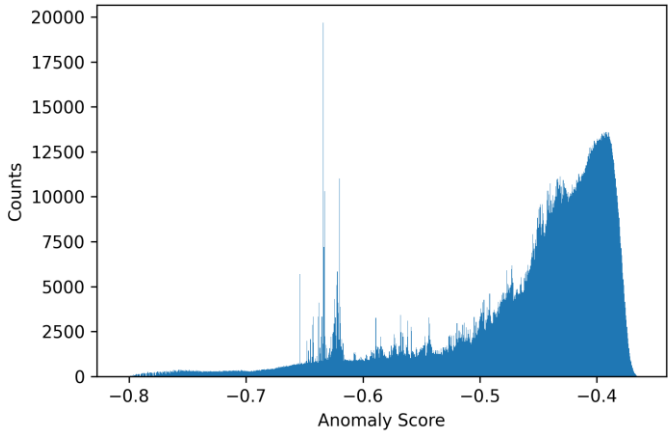


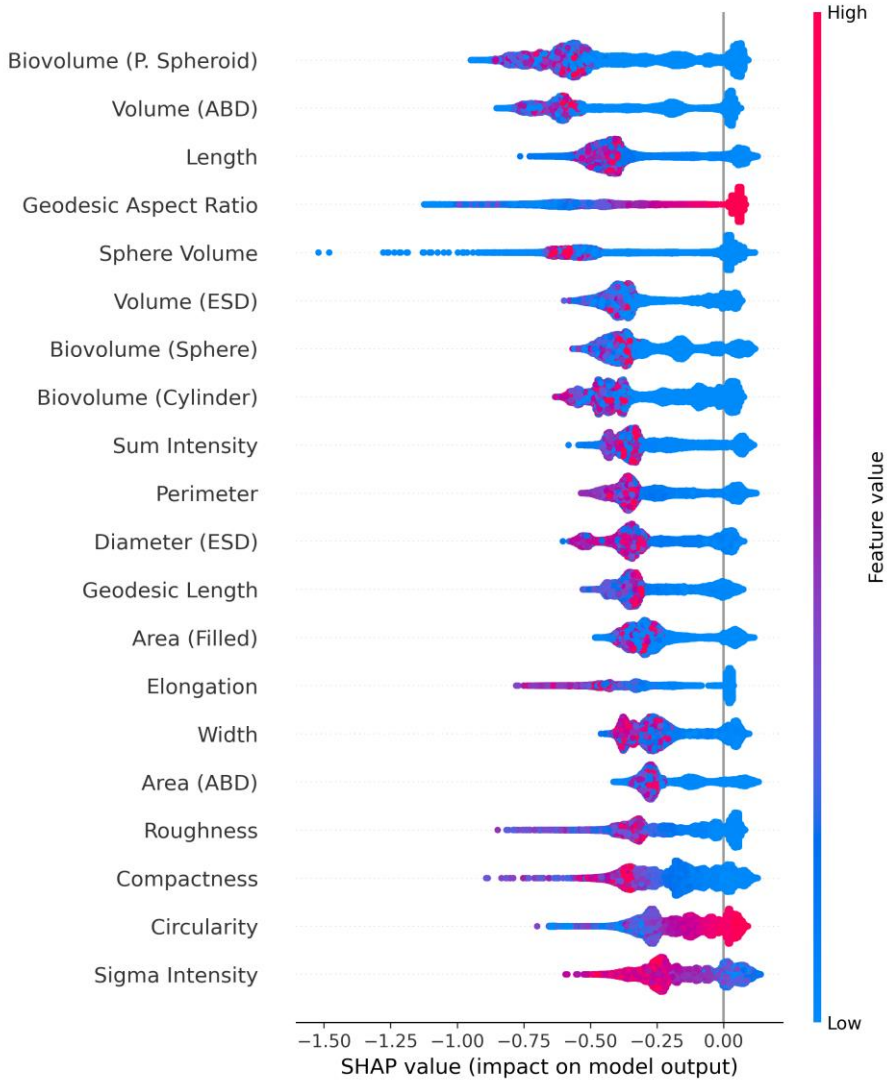
Results | Metadata and images

Isolation Forest Scores Synthetic



Isolation Forest Scores All Ice Data



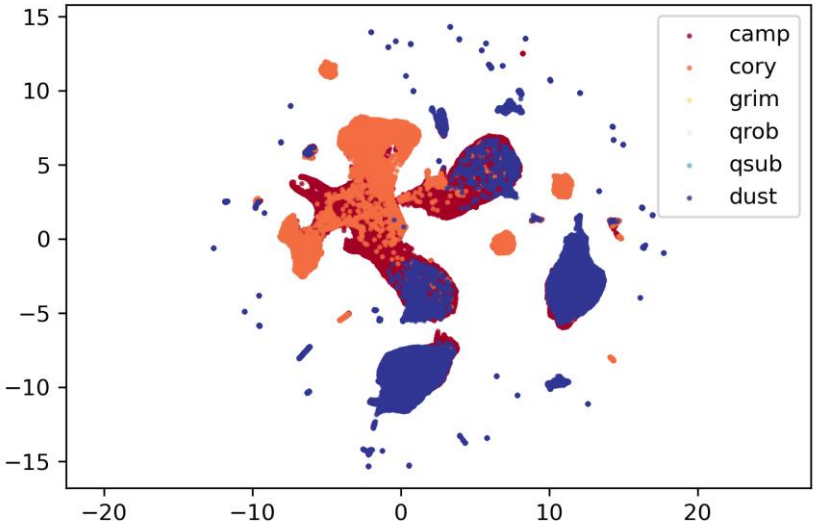


Unsupervised learning (UMAP)

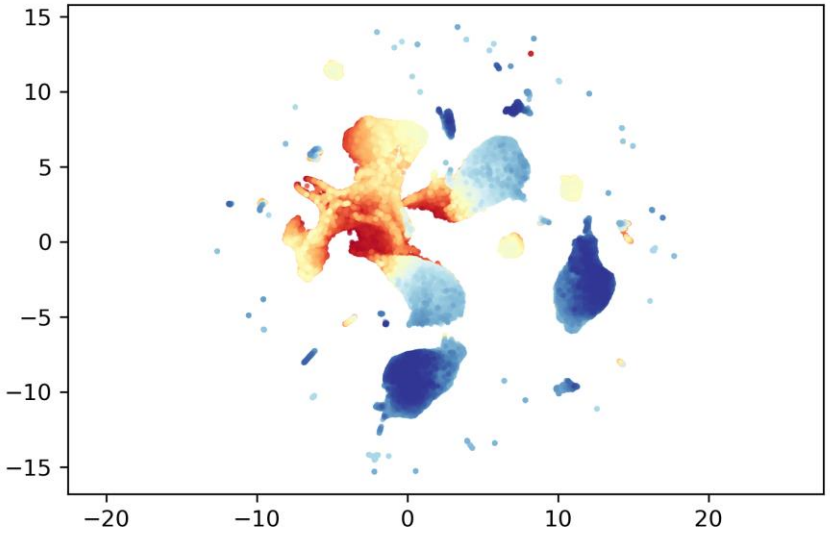
Results | Unsupervised learning (UMAP)

- Synthetic data

UMAP Synthetic Data: Class



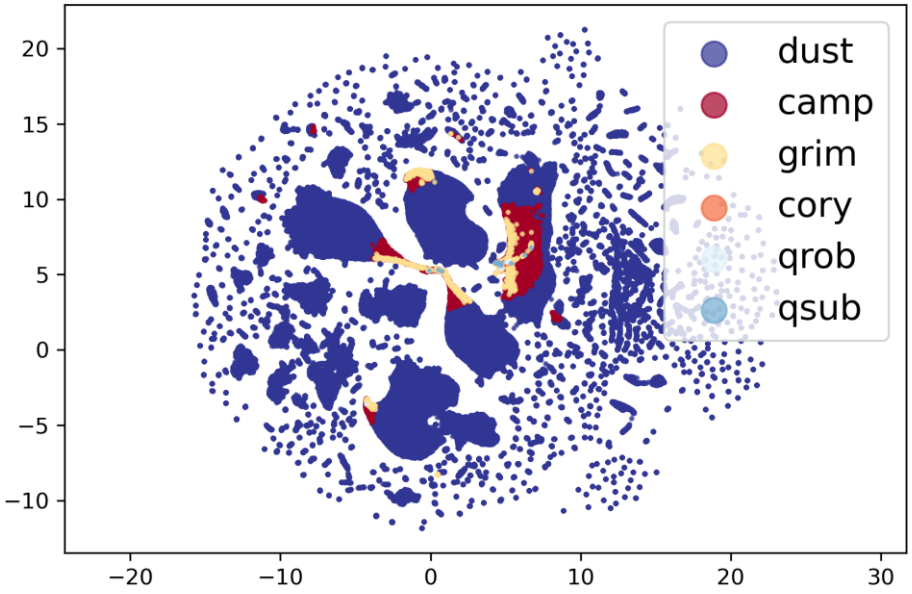
UMAP Synthetic Data: Anomaly



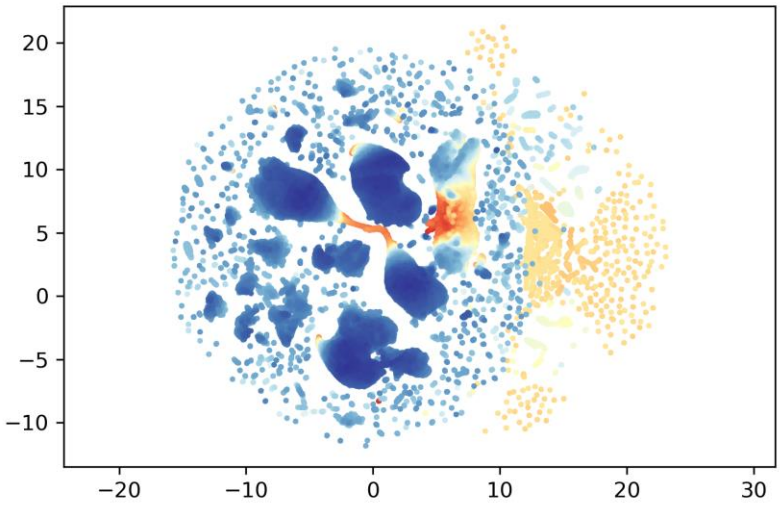
Results | Unsupervised learning (UMAP)

- Real ice data

UMAP 15% Total Ice Data:
Colored by Predicted Class



UMAP 15% Total Ice Data:
Colored by Anomaly Score



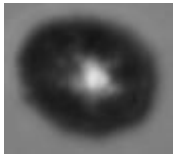
Future work

- Investigate more elaborate CNN architectures
- Experimenting more with data-augmentation
- Automated HP optimisation
- Auto encoders
- One-shot learning

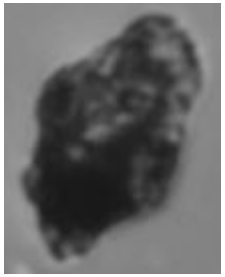
Conclusion

- Synthetic ice core insolubles identified to a relatively high degree: up to 99%
- Utilising meta-data only; very efficient
- Anomalies detected in both synthetic data and real data
- Structures in the real ice core data were found using unsupervised learning.

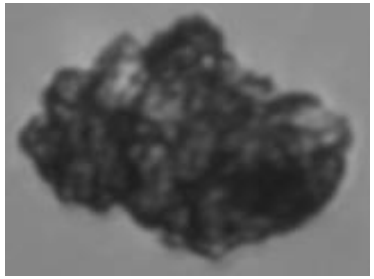
Six interesting anomalous images from GRIP



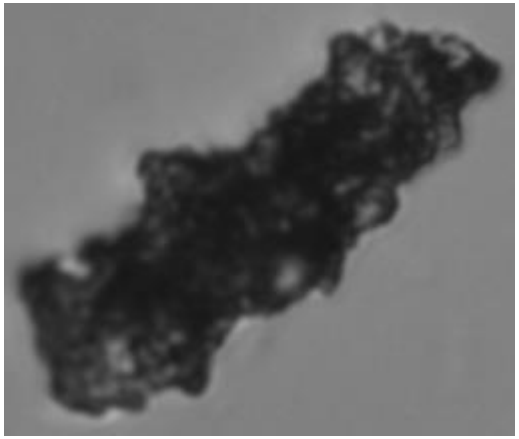
GRIP_3046_40_55_1_2787



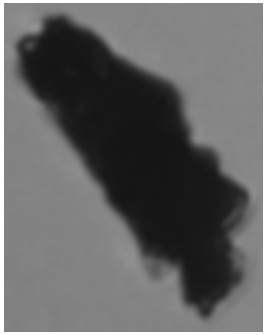
GRIP_3136_0_20_1_828



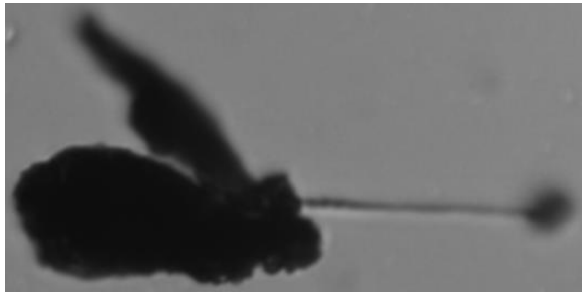
GRIP_3046_40_55_1_12275



GRIP_3136_0_20_1_823



GRIP_3136_0_20_1_830



GRIP_3136_0_20_1_805

References

- He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015): 1904-1916.
- Yosinski, Jason, et al. "How transferable are features in deep neural networks?." *arXiv preprint arXiv:1411.1792* (2014).
- He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

References

- Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians." *Journal of the American statistical Association* 112.518 (2017): 859-877.
- Koushik, Jayanth. "Understanding convolutional neural networks." *arXiv preprint arXiv:1605.09081* (2016).
- Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." *ICML deep learning workshop*. Vol. 2. 2015.
- LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

Reference

- Kuo, C-C. Jay. "Understanding convolutional neural networks with a mathematical model." *Journal of Visual Communication and Image Representation* 41 (2016): 406-413.
- Aizman, Alex, Gavin Maltby, and Thomas Breuel. "High Performance I/O For Large Scale Deep Learning." *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019.

Appendix A.2

Pre-processing, data augmentation, setup | CNNs

```
TRAIN_TRANSFORM = tv_transforms.Compose([
    tv_transforms.RandomApply([
        tv_transforms.RandomCrop(80, pad_if_needed=True)],
        p=0.10),
    tv_transforms.Resize([IMAGE_WIDTH, IMAGE_HEIGHT],
        interpolation=tv_transforms.InterpolationMode.BILINEAR),
    tv_transforms.Normalize(mean=[60], std=[30], inplace=True),
    tv_transforms.RandomHorizontalFlip(p=0.10),
    tv_transforms.RandomVerticalFlip(p=0.10),
    tv_transforms.RandomApply([tv_transforms.GaussianBlur(5)], p=0.10)
])
```

Appendix A.2

Pre-processing, data augmentation, setup | CNNs

- Evaluation (val: 15%, test: 15%)
- Loss:
  - PyTorch: CrossEntropyLoss
  - Keras: SparseCategorialCrossEntropyLoss
- Optimizer: Adam
- Learning rate:
  - PyTorch: 0.001
  - Keras: 0.0008

Appendix B.1 | PyTorch architectures

Architecture 1

```
----------------------------------------------------------------
   Layer (type)               Output Shape         Param #
================================================================
      Conv2d-1         [-1, 8, 118, 118]             976
     MaxPool2d-2          [-1, 8, 59, 59]               0
      Conv2d-3         [-1, 16, 55, 55]           3,216
     MaxPool2d-4         [-1, 16, 27, 27]               0
      Linear-5              [-1, 108]           1,259,820
      Linear-6                [-1, 6]                 654
================================================================
Total params: 1,264,666
Trainable params: 1,264,666
Non-trainable params: 0
Input size (MB): 0.06
Forward/backward pass size (MB): 1.52
Params size (MB): 4.82
Estimated Total Size (MB): 6.41
----------------------------------------------------------------
```

Appendix B.2 | PyTorch architectures

Architecture 2

```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1         [-1, 8, 118, 118]             976
       MaxPool2d-2           [-1, 8, 59, 59]               0
          Conv2d-3          [-1, 16, 55, 55]           3,216
       MaxPool2d-4          [-1, 16, 27, 27]               0
          Linear-5                [-1, 108]       1,259,820
     BatchNorm1d-6                [-1, 108]             216
          Dropout-7                [-1, 108]               0
          Linear-8                  [-1, 6]             654
================================================================
Total params: 1,264,882
Trainable params: 1,264,882
Non-trainable params: 0
Input size (MB): 0.06
Forward/backward pass size (MB): 1.52
Params size (MB): 4.83
Estimated Total Size (MB): 6.41
----------------------------------------------------------------
```

Appendix B.3 | PyTorch architectures

Architecture 3

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 8, 118, 118]	976
MaxPool2d-2	[-1, 8, 59, 59]	0
Conv2d-3	[-1, 16, 55, 55]	3,216
MaxPool2d-4	[-1, 16, 27, 27]	0
Conv2d-5	[-1, 32, 25, 25]	4,640
MaxPool2d-6	[-1, 32, 12, 12]	0
Linear-7	[-1, 64]	294,976
BatchNorm1d-8	[-1, 64]	128
Dropout-9	[-1, 64]	0
Linear-10	[-1, 6]	390

Total params: 304,326
Trainable params: 304,326
Non-trainable params: 0

Input size (MB): 0.06
Forward/backward pass size (MB): 1.71
Params size (MB): 1.16
Estimated Total Size (MB): 2.93

Appendix B.4 | PyTorch architectures

Architecture 4

```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1         [-1, 8, 118, 118]             976
       MaxPool2d-2           [-1, 8, 59, 59]               0
          Conv2d-3          [-1, 16, 55, 55]           3,216
       MaxPool2d-4          [-1, 16, 27, 27]               0
          Conv2d-5          [-1, 32, 25, 25]           4,640
       MaxPool2d-6          [-1, 32, 12, 12]               0
          Conv2d-7          [-1, 64, 10, 10]          18,496
       MaxPool2d-8           [-1, 64, 5, 5]               0
          Linear-9                 [-1, 40]          64,040
     BatchNorm1d-10                 [-1, 40]              80
         Dropout-11                 [-1, 40]               0
         Linear-12                  [-1, 3]             123
================================================================
Total params: 91,571
Trainable params: 91,571
Non-trainable params: 0
Input size (MB): 0.06
Forward/backward pass size (MB): 1.77
Params size (MB): 0.35
Estimated Total Size (MB): 2.18
```

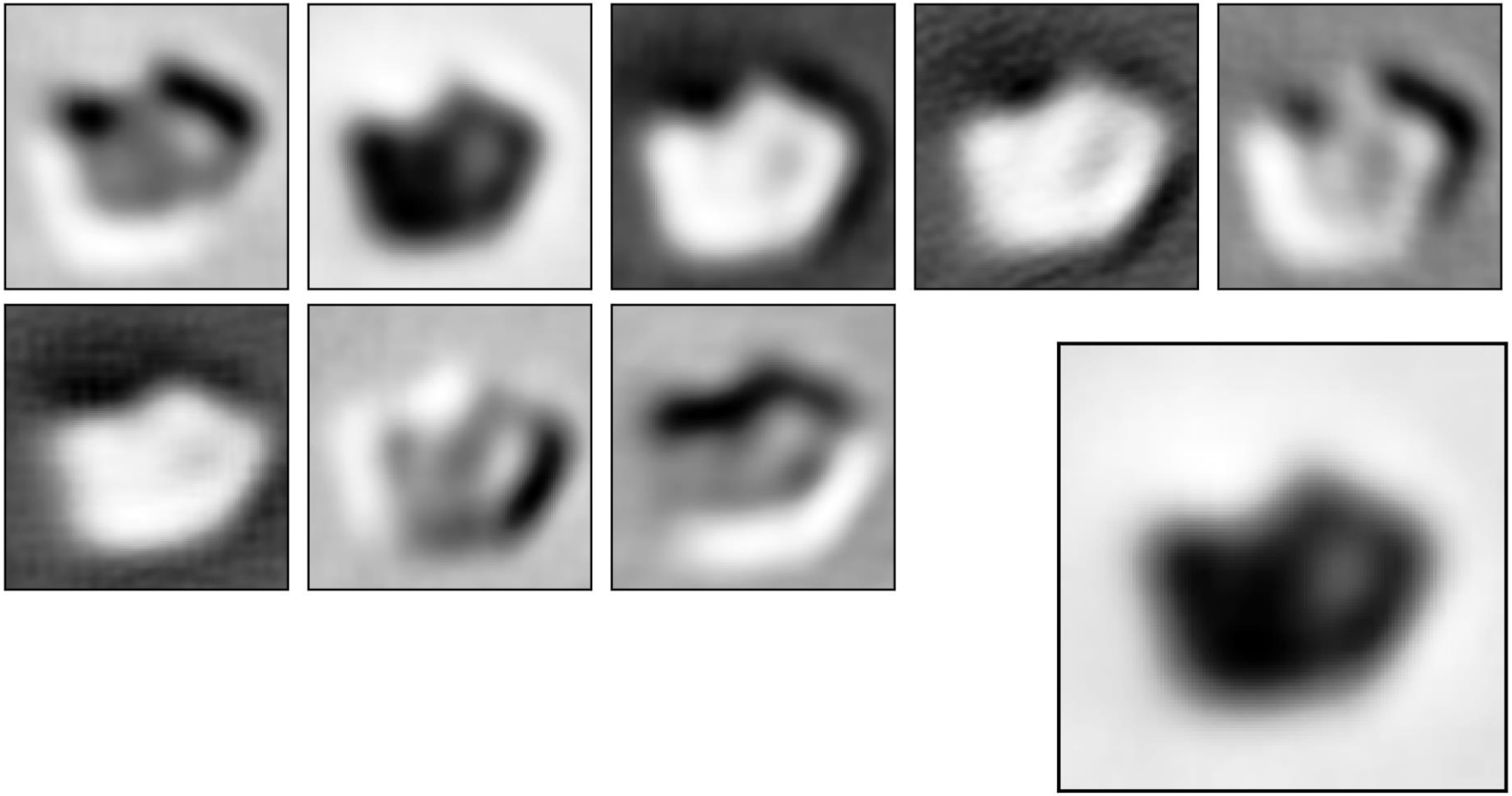
Appendix B.5 | Keras architecture

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 128, 128, 1)	0
conv2d (Conv2D)	(None, 126, 126, 8)	80
max_pooling2d (MaxPooling2D)	(None, 63, 63, 8)	0
conv2d_1 (Conv2D)	(None, 61, 61, 16)	1168
max_pooling2d_1 (MaxPooling2	(None, 30, 30, 16)	0
conv2d_2 (Conv2D)	(None, 28, 28, 32)	4640
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295040
dense_1 (Dense)	(None, 3)	387

Total params: 319,811
Trainable params: 319,811
Non-trainable params: 0

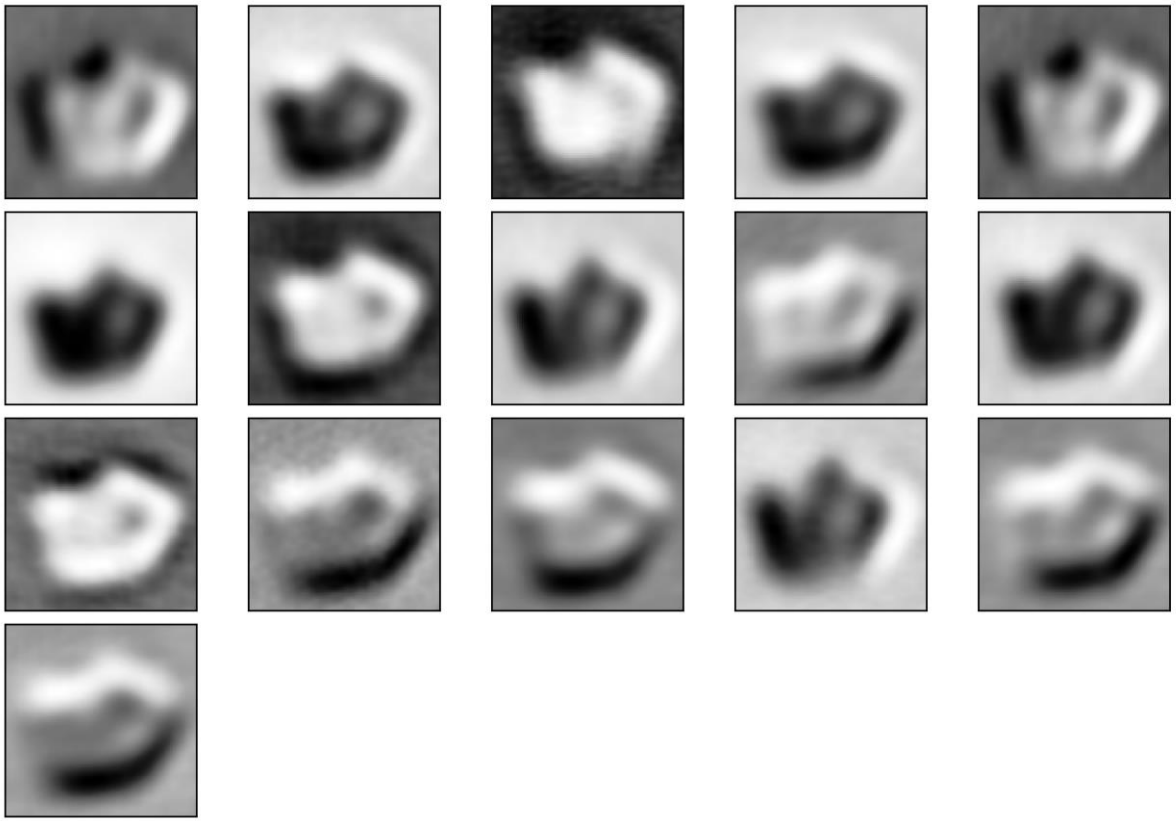
Appendix C.1 | Feature maps

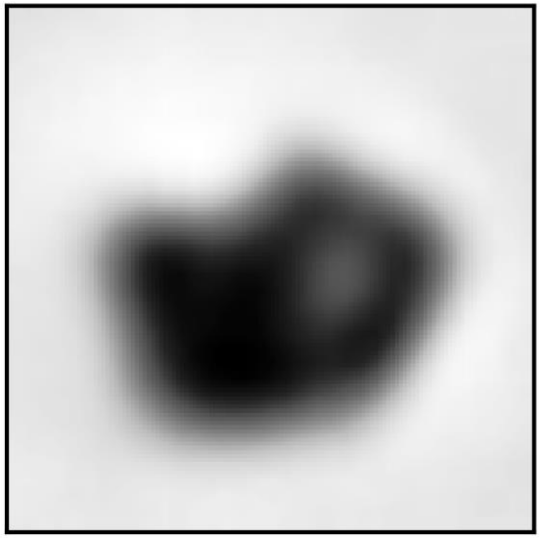
- PyTorch A3 on 6 class problem, Conv2d-1



Appendix C.2 | Feature maps

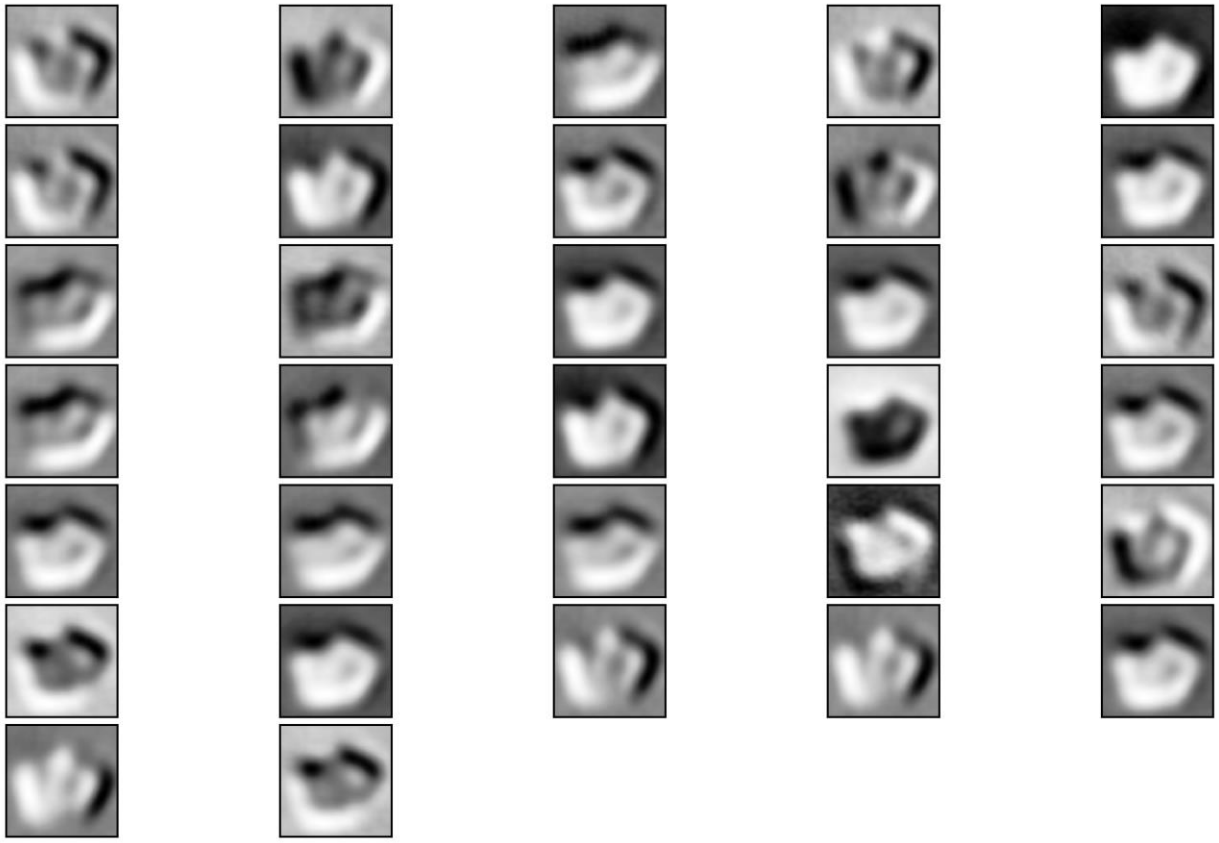
- PyTorch A3 on 6 class problem, Conv2d-2

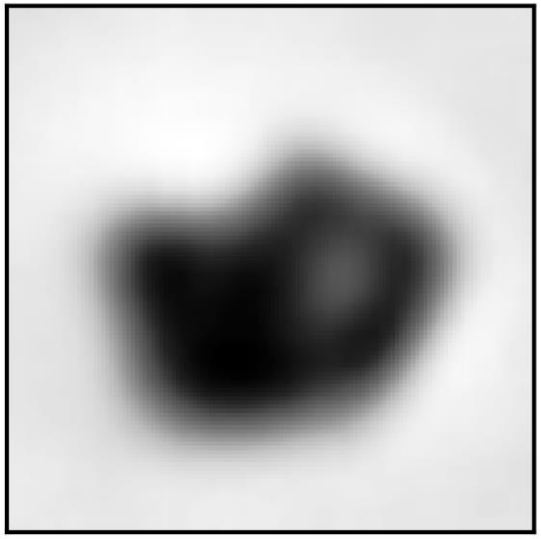




Appendix C.3 | Feature maps

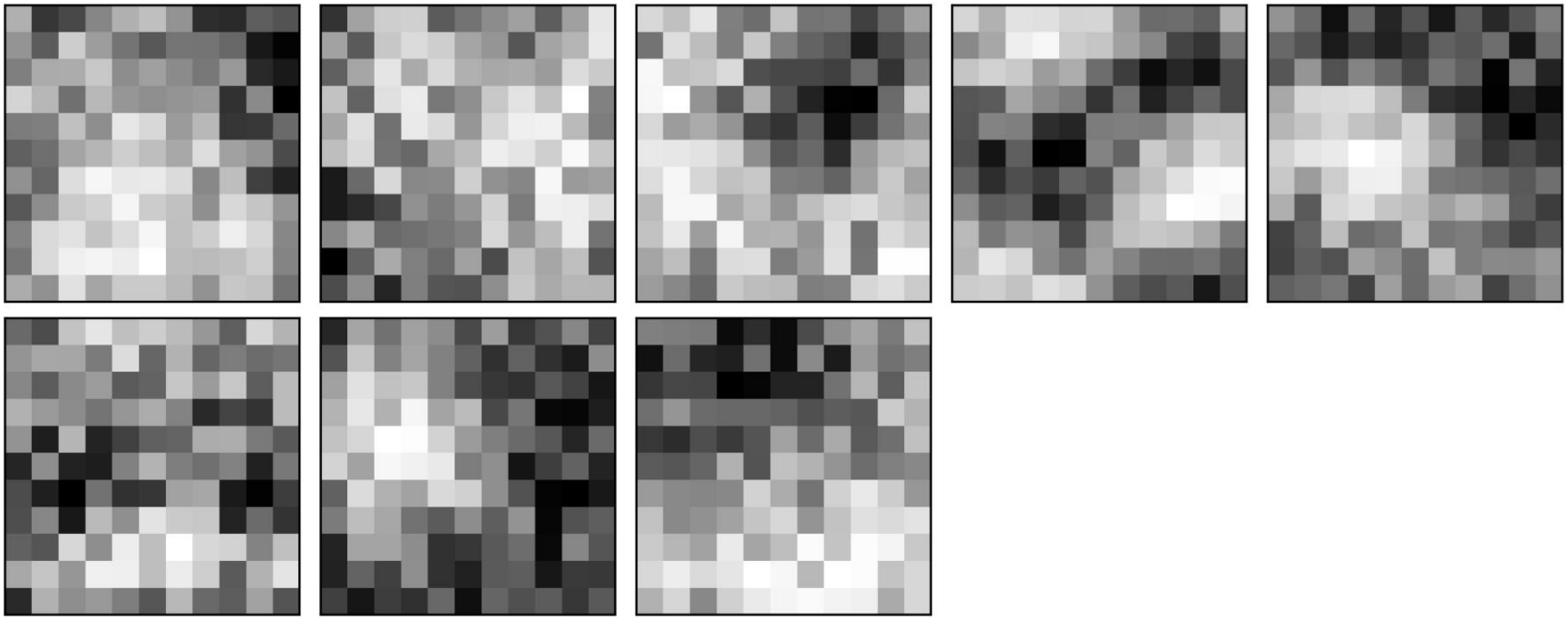
- PyTorch A3 on 6 class problem, Conv2d-3





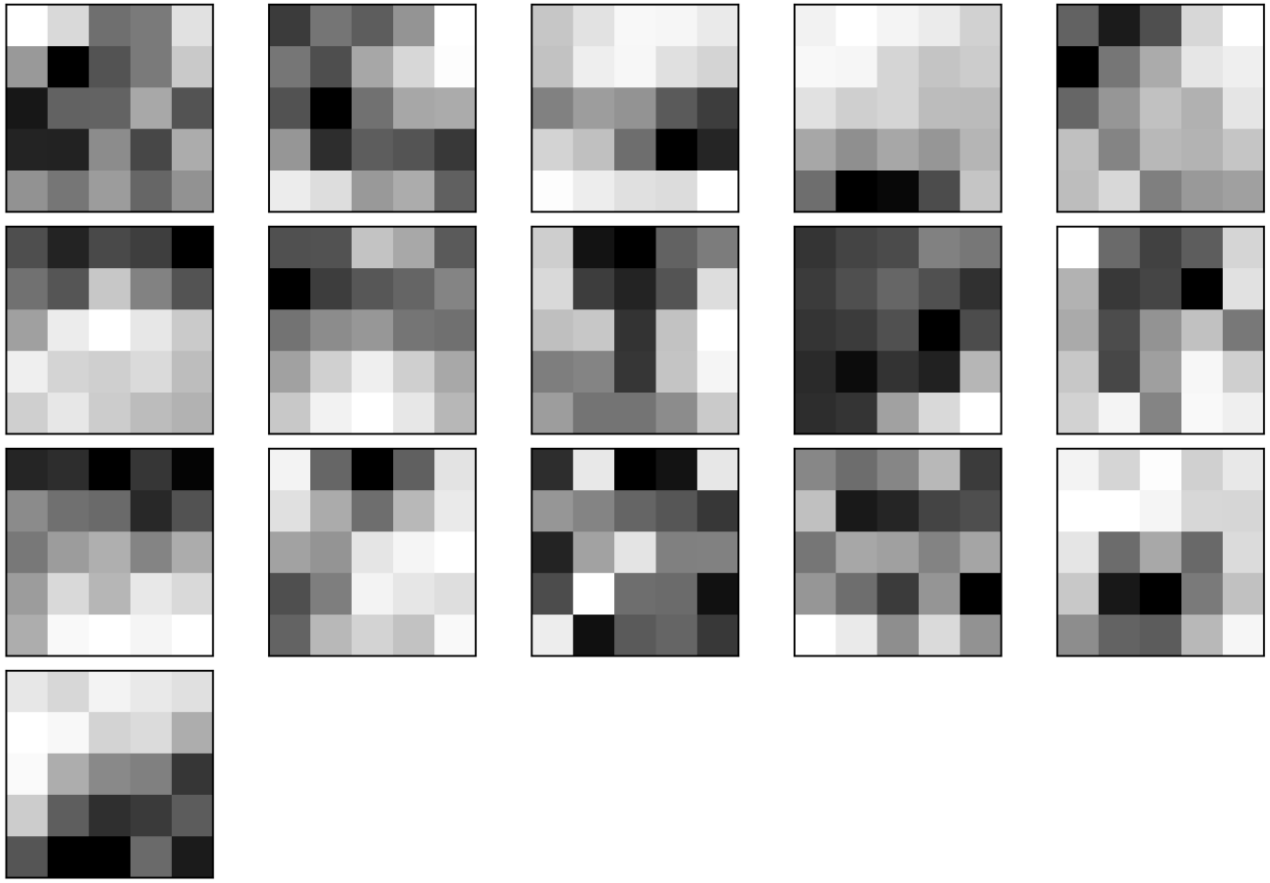
Appendix D.1 | Filters

- PyTorch A3 on 6 class problem, Conv2d-1



Appendix D.2 | Filters

- PyTorch A3 on 6 class problem, Conv2d-2

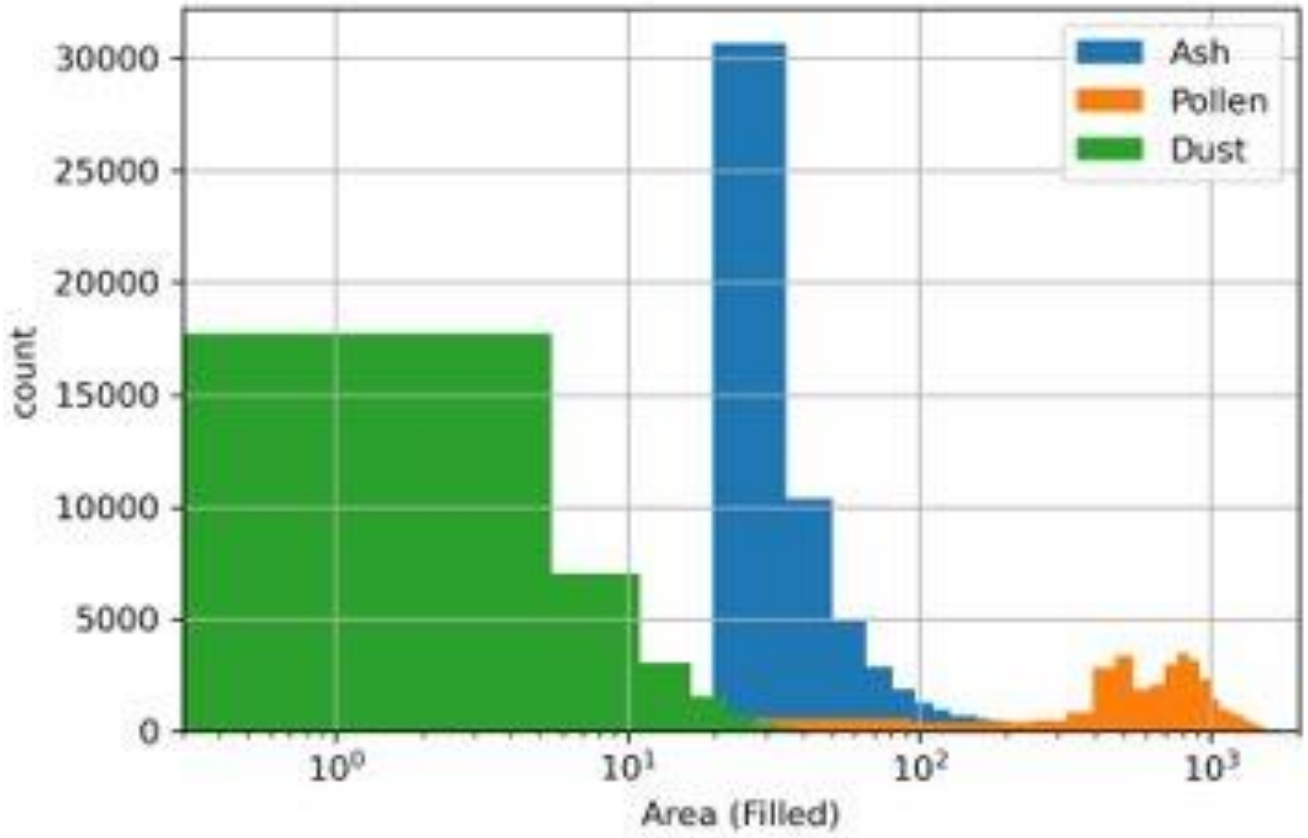


Appendix D.3 | Filters

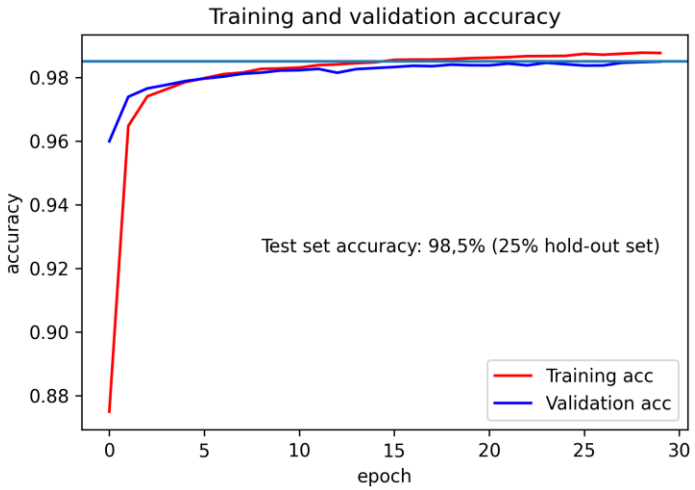
- PyTorch A3 on 6 class problem, Conv2d-3

Appendix E | Synthetic data size distributions

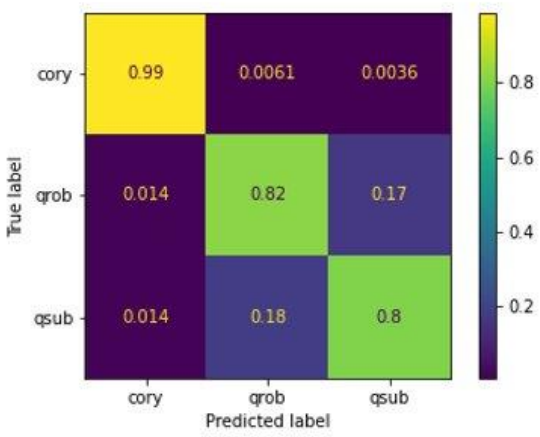
UMAP ICE Data: fit_transform ICE

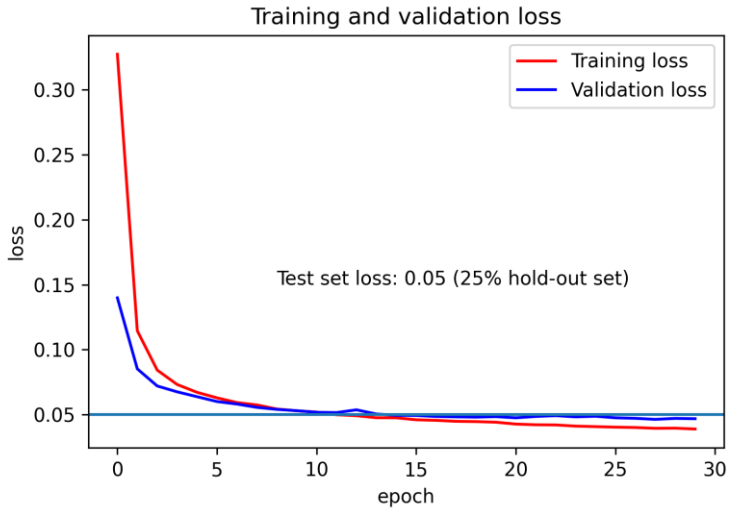


Appendix F | Classification: Transfer Learning Only Keras Dense Feedforward NN Classifier



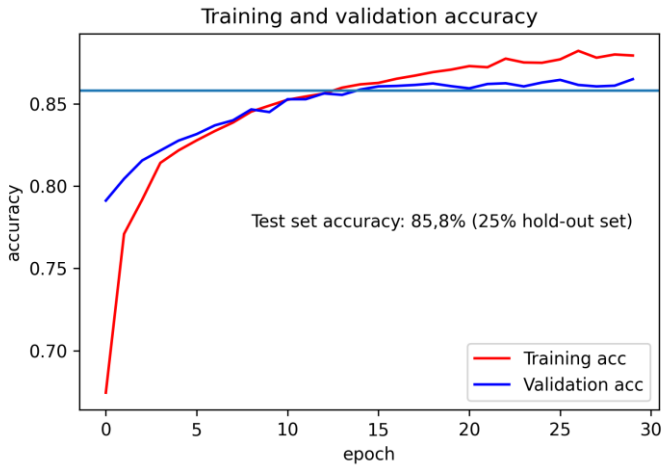
3 Class

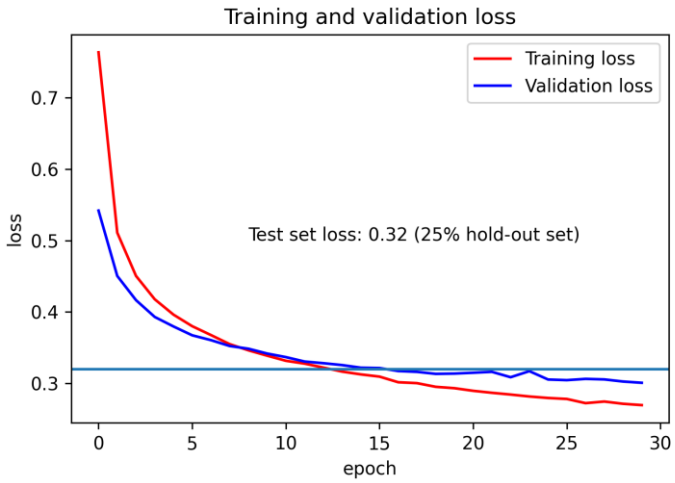




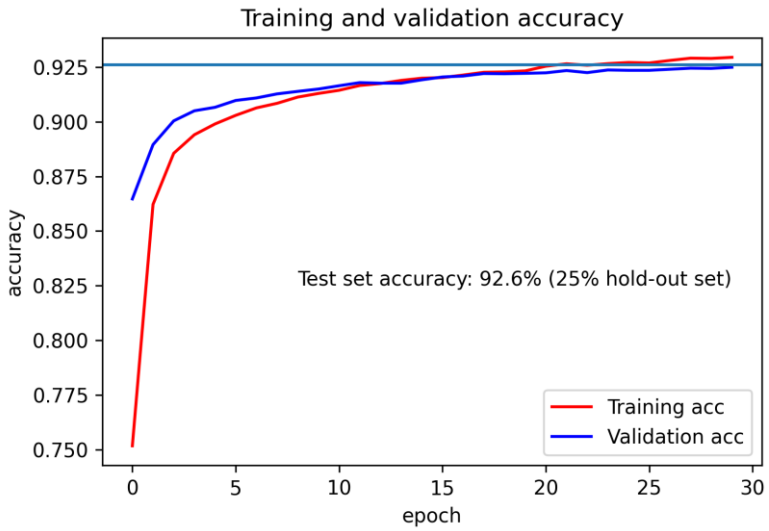
Appendix H | Classification: Transfer Learning Only Keras Dense Feedforward NN Classifier

Pollen Only

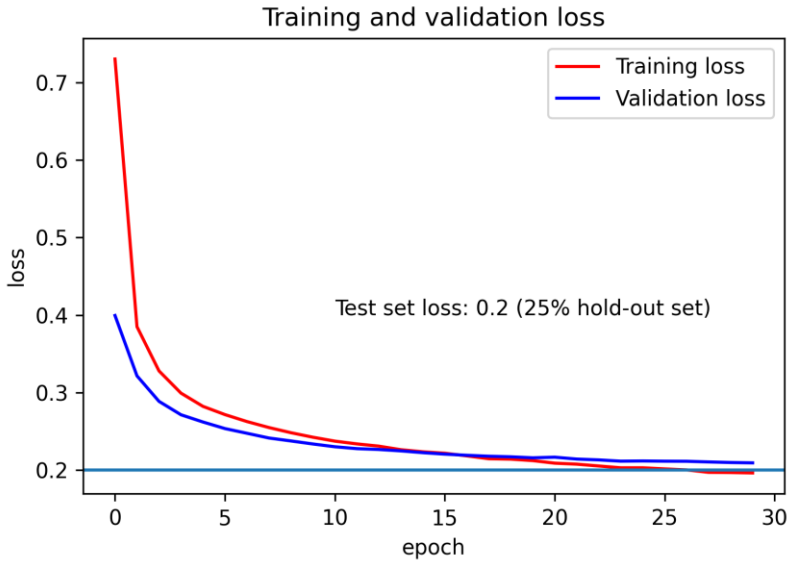




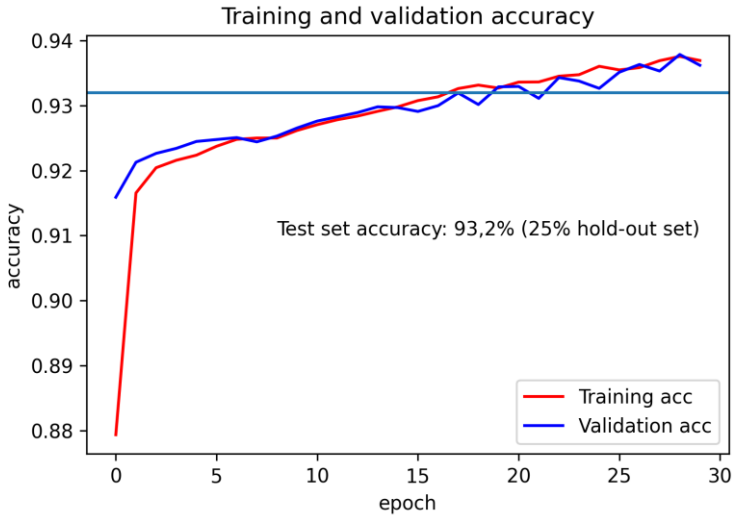
Appendix G | Classification: MetaData and Transfer Learning Keras Dense Feedforward NN Classifier



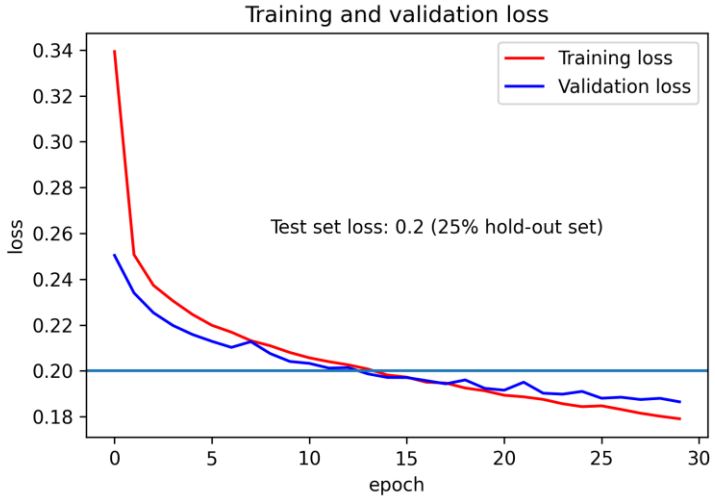
6 class classification



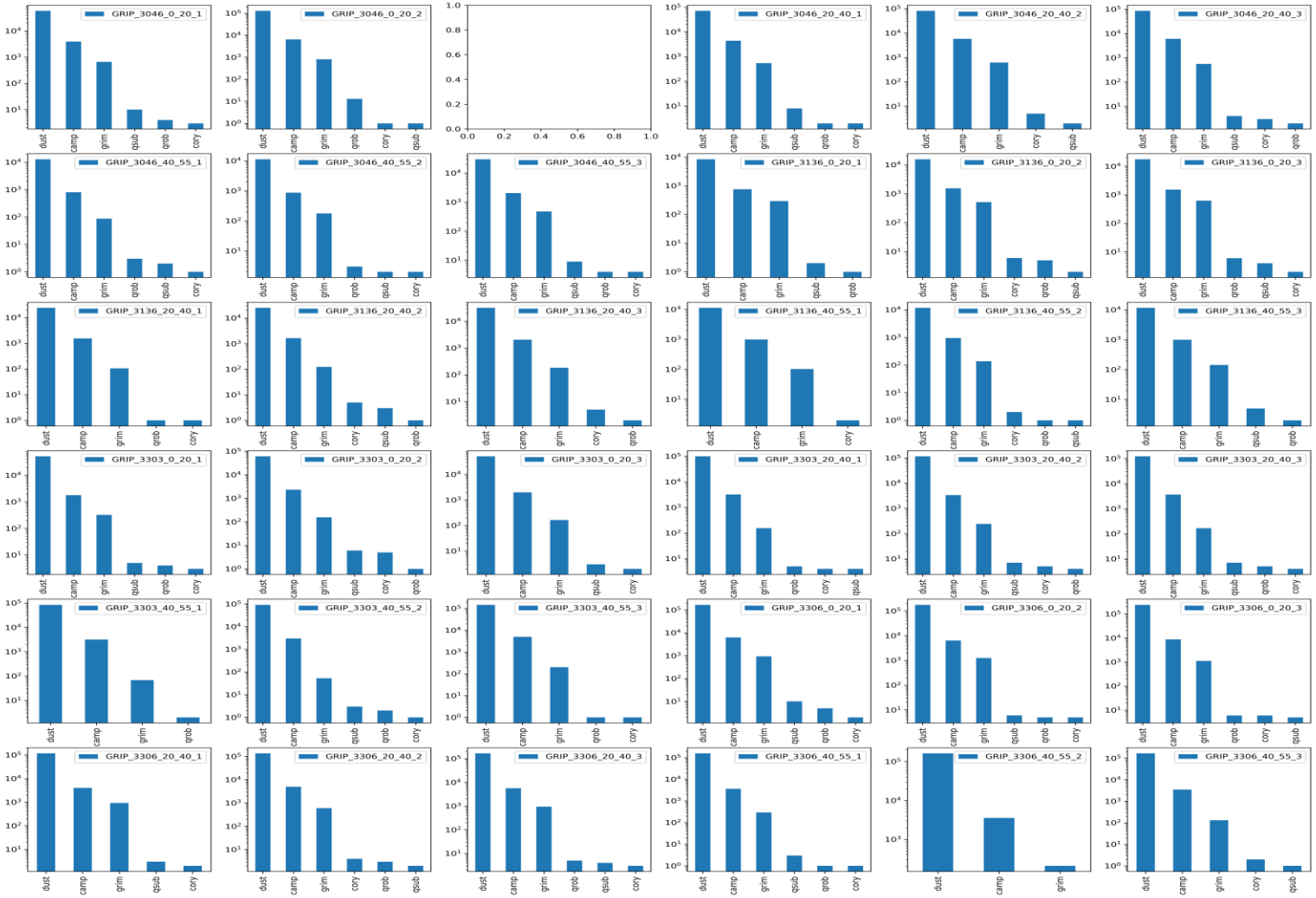
Appendix I | Classification: Transfer Learning Only Keras Dense Feedforward NN Classifier



Ash Only



Appendix I.1 | Predicted Class Distributions for real ICE data set by sample, segment, and repeated measurement



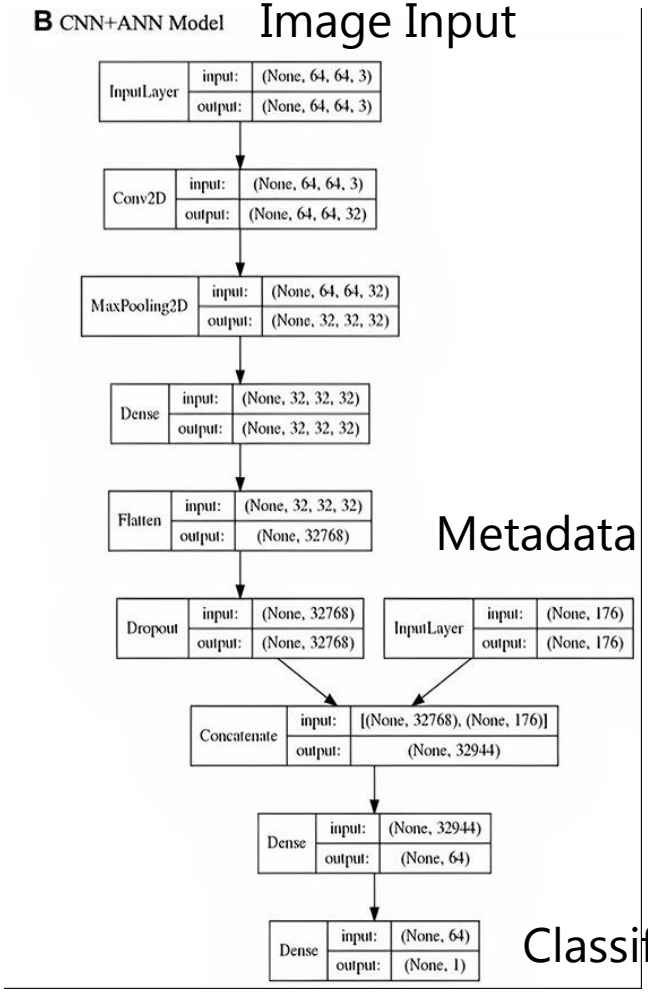
Appendix J | Also investigated

- Autoencoders – vanilla, convolutional, variational
  - Autoencoders may be used as anomaly detection
  - Convolutional autoencoders are appropriate for use with Images
  - Variational autoencoders may be used for a rich form of data augmentation

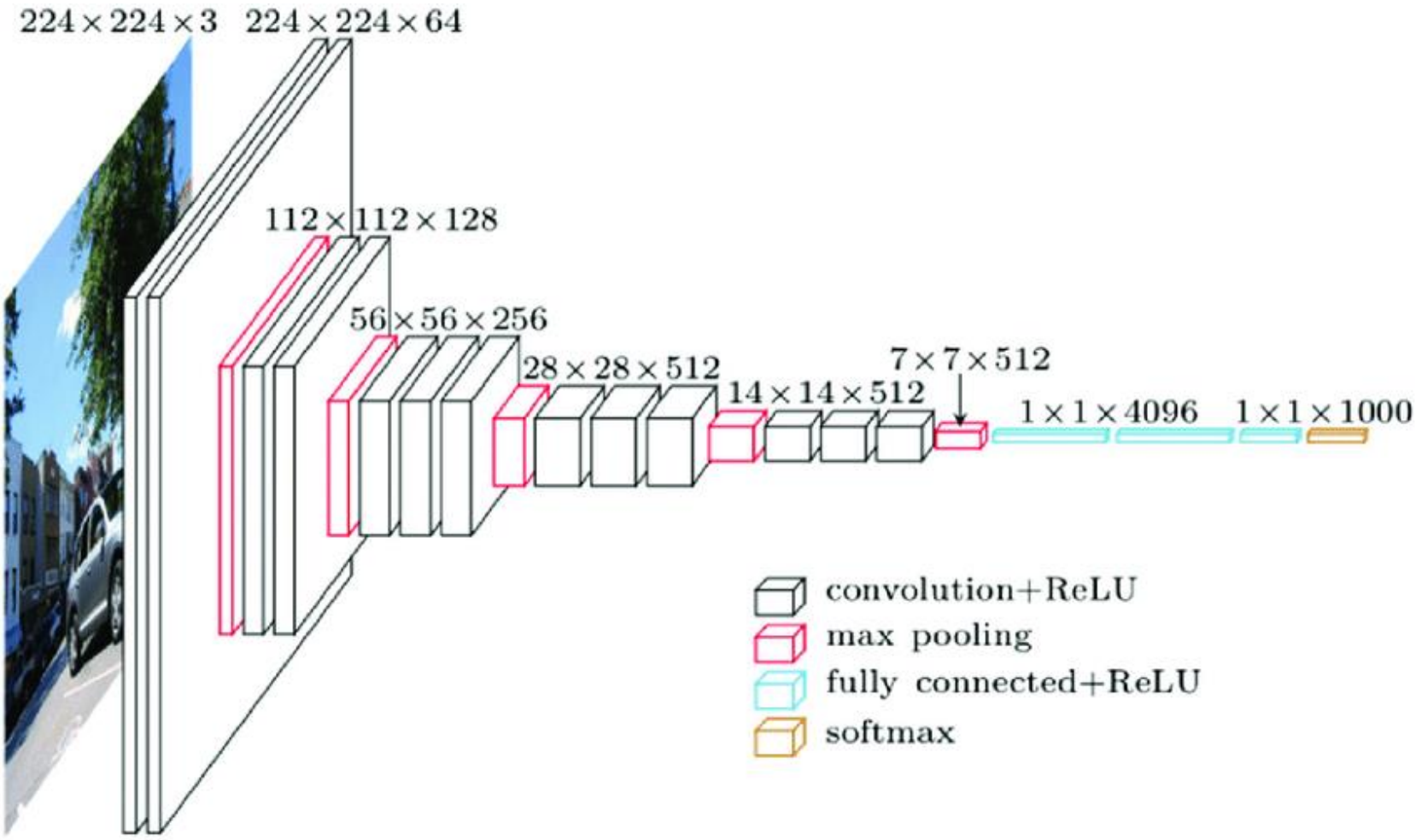
- Zero, one, few, and many shot learning
  - Use a siamese network to train on pairs of classes allowing classification to be performed on a similarity metric to high accuracy with very few if any labelled classes!

Appendix K | Schematic combining image data with metadata

Ningrum, Dina Nur Anggraini, et al. "Deep Learning Classifier with Patient's Metadata of Dermoscopic Images in Malignant Melanoma Detection." *Journal of Multidisciplinary Healthcare* 14 (2021): 877.



Appendix L | VGG Architecture



Appendix M | Isolation Forest: Separating Power