Skin Lesions

KU - Big data Analysis 2020

Helena Britze
Katja Johansen
Simon Hilbard
Viktoria Lavro

Equal team effort

Goal of the project

Based on the images and their labels,
the purpose of the project is to classify skin lesions.

Overview

- Understanding the **Data**
- Machine Learning **Algorithm**
- **Applying** the Algorithm before/less processing of the data
- **Data processing**
- **Optimizing**
- Evaluation of the **Results**
- **Problems**
- **Conclusion**

Data

Cancerous groups

- *akiec:* actinic keratoses and intraepithelial carcinoma / Bowen's disease
- *bcc:* basal cell carcinoma
- *mel:* melanoma

Benign groups

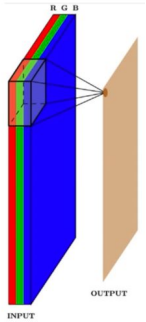
- *bkl:* benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses)
- *df:* dermatofibroma
- *nv*: melanocytic nevi
- *vasc:* vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage

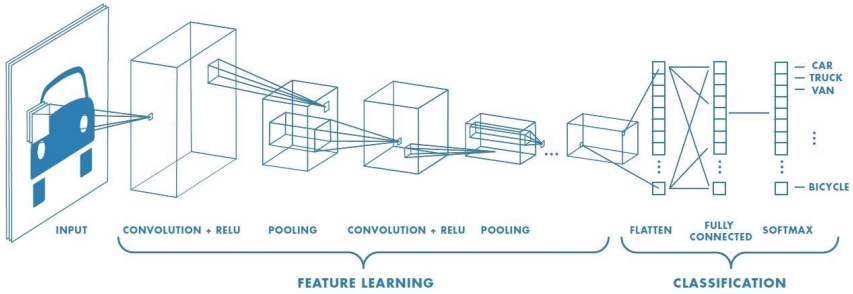
HARVARD Dataverse

Machine Learning algorithm

The Machine Learning algorithm that the project is build on is the **Convolutional Neural Network**.

We choose to use a **3D CNN**, because the color seems to be an imported factor for classifying the skin lesions.



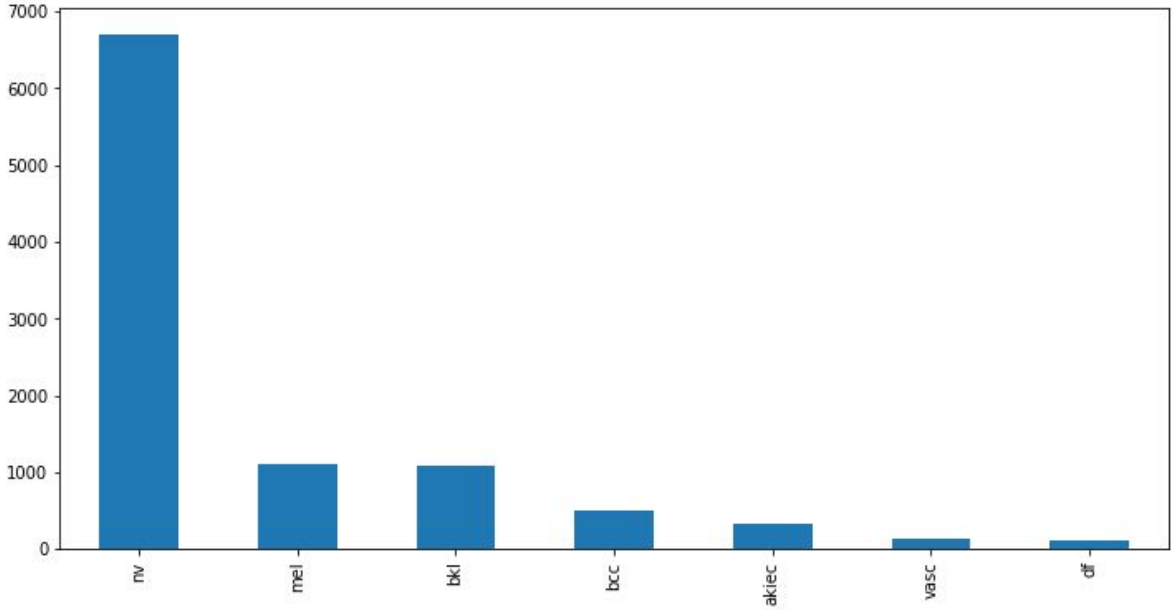


Data structure

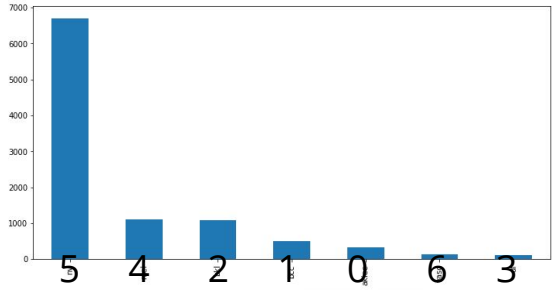
Distribution of the **10.015** images, based on seven cell types.

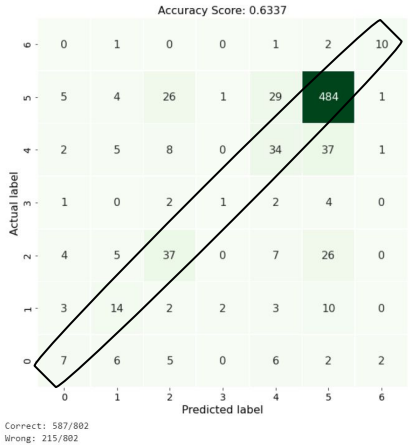
Furthermore we have some **metadata** on the patients. We will get back to that.

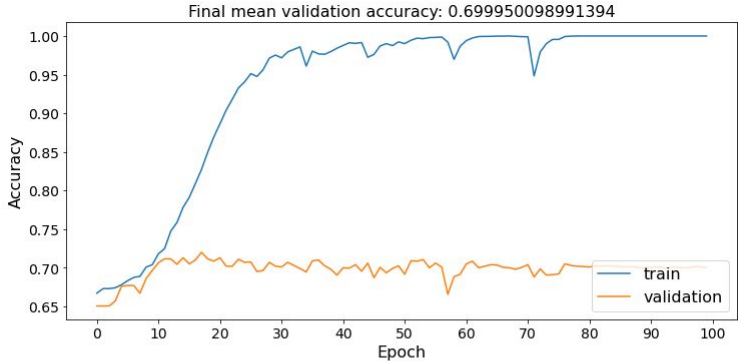
More than **95%** of all lesion during clinical practice will fall into one of the seven categories [1].

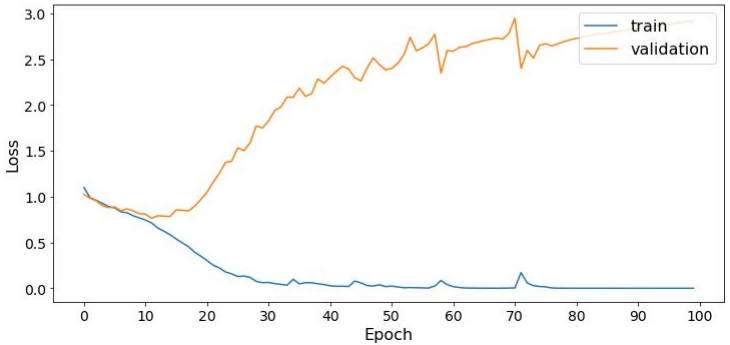


CNN performers without data processing



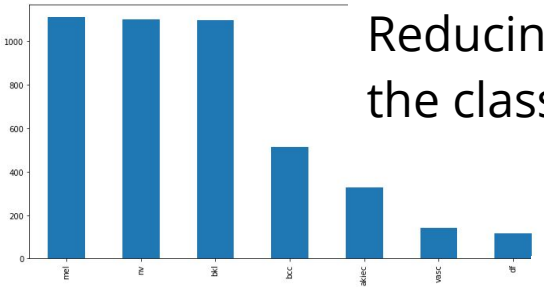




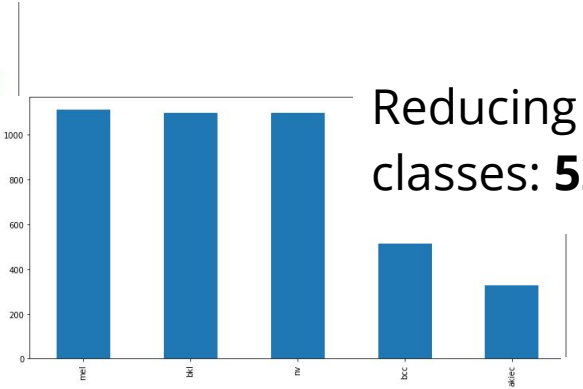


68.5 % accuray

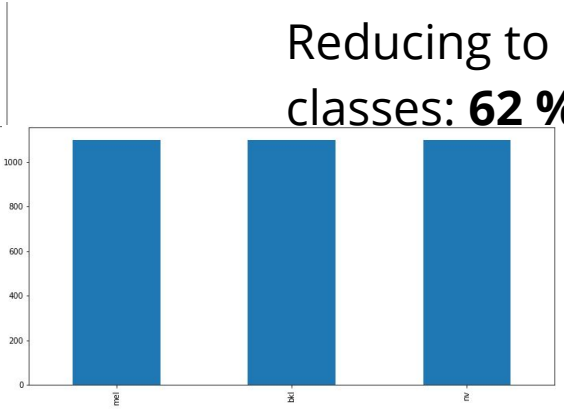
CNN performance, simple data processing



Reducing one of the classes: **52 %**

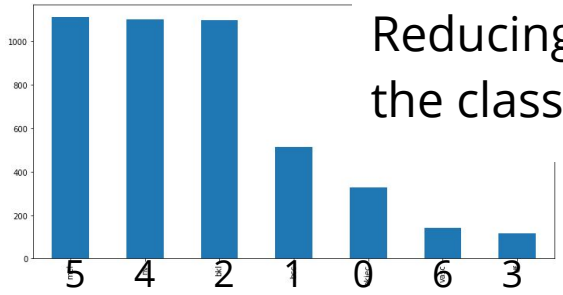


Reducing to 5 classes: **52 %**

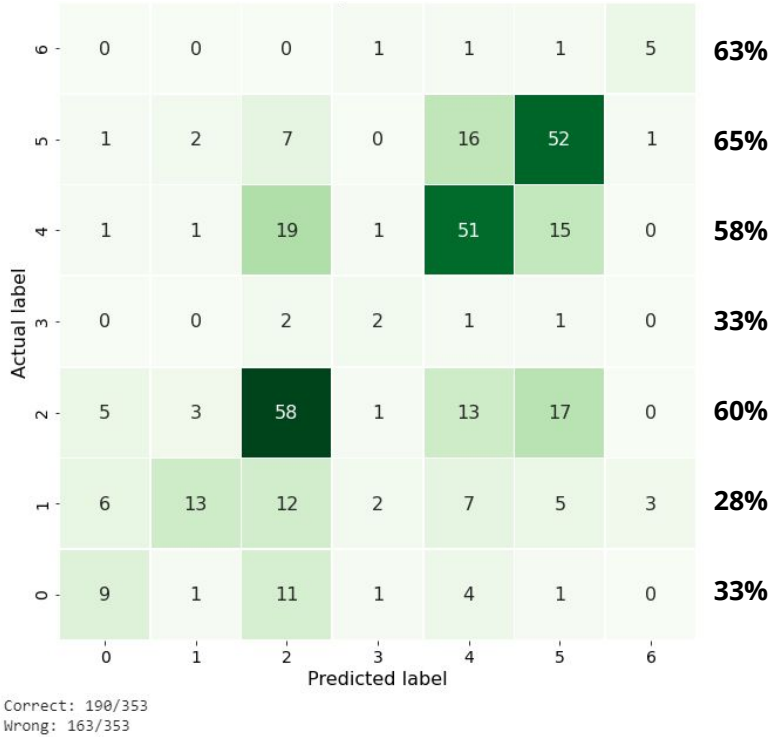


Reducing to 3 classes: **62 %**

CNN performance, simple data processing



Reducing one of the classes: **52 %**



Inclusion of metadata

We have information on **age**, **sex** and **localization** (on the body) from the patients, which could improve learning.
For example:

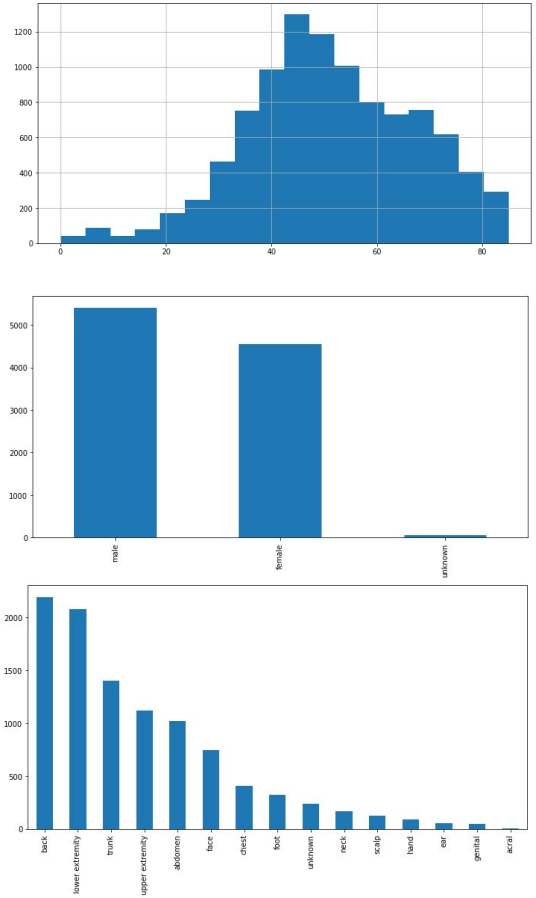
Expect more hair if:

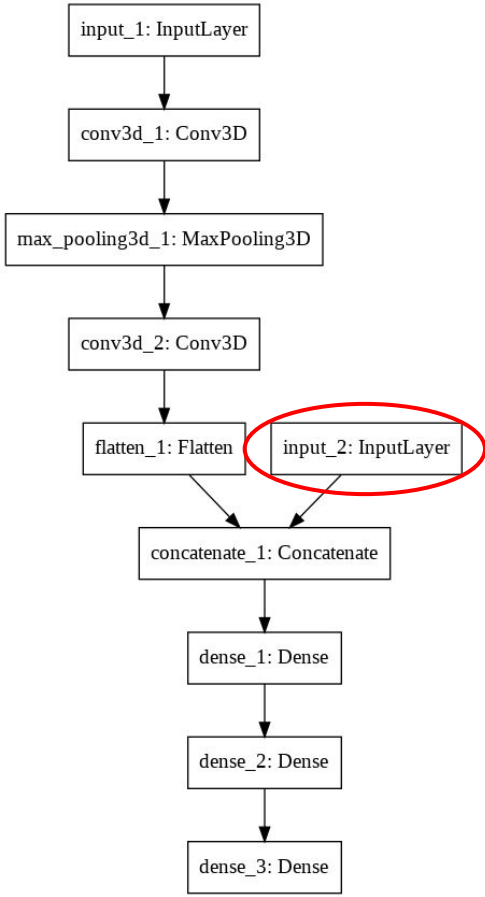
- sex = male
- localization = eg. scalp

Expect more uneven skin if:

- age is higher

We had to filter out unknowns.





We feed in the metadata only after the convolutional layers

Inclusion of metadata

We have information on **age**, **sex** and **localization** (on the body) from the patients, which could improve learning.
For example:

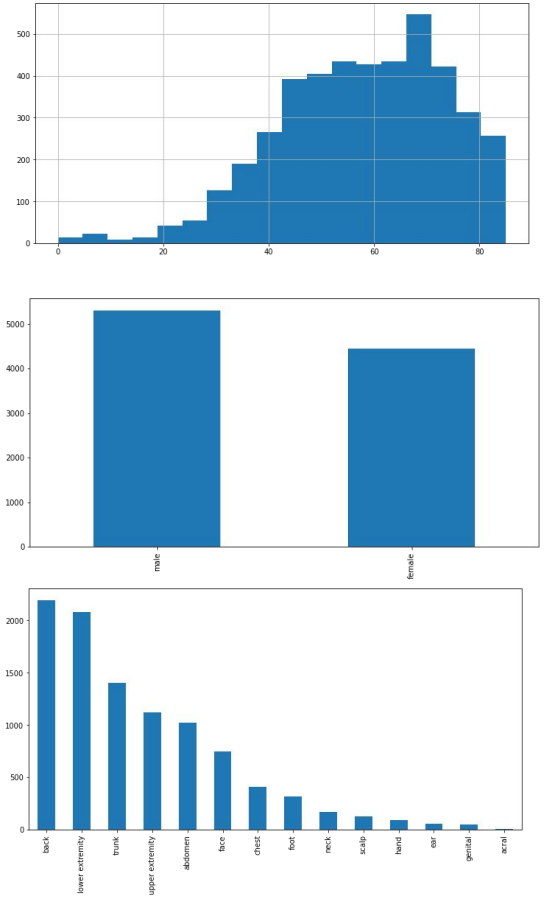
Expect more hair if:

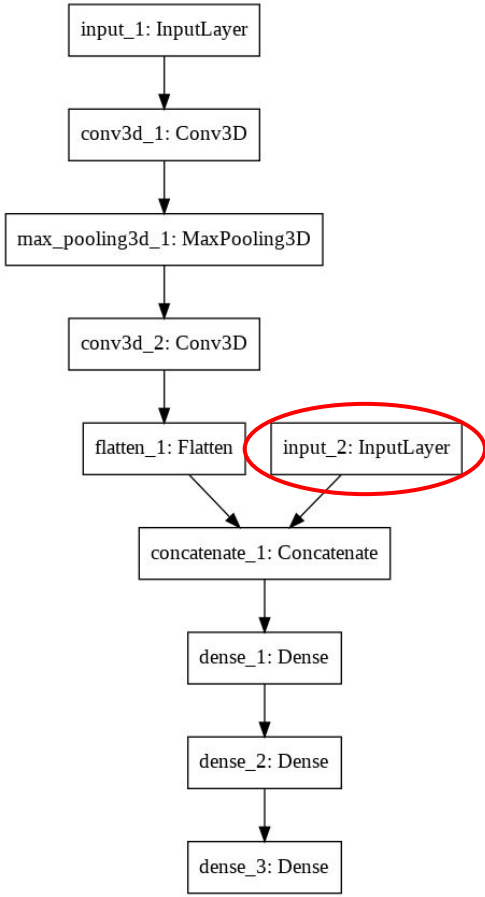
- sex = male
- localization = eg. scalp

Expect more uneven skin if:

- age is higher

We had to filter out unknowns.

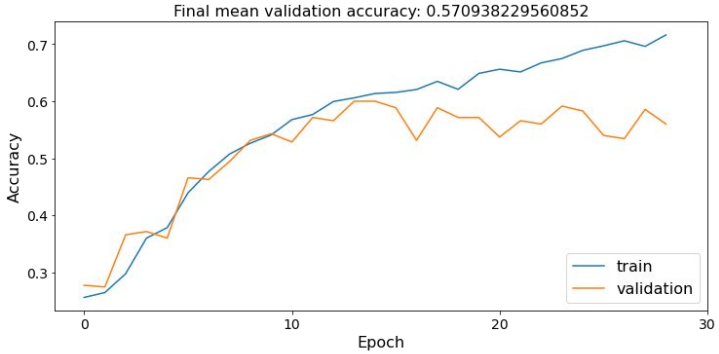


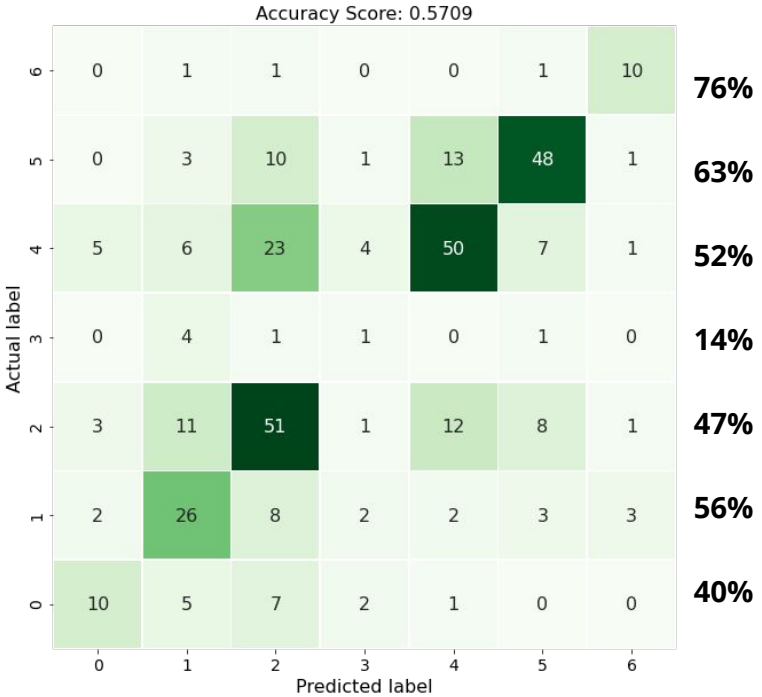


We feed in the metadata only after the convolutional layers

Inclusion of metadata

Test accuracy improvement from 52% to **57%**





How can we overcome the unbalanced dataset?

We can optimize the network and obtain a reasonable accuracy, but in return we remove most of the images.

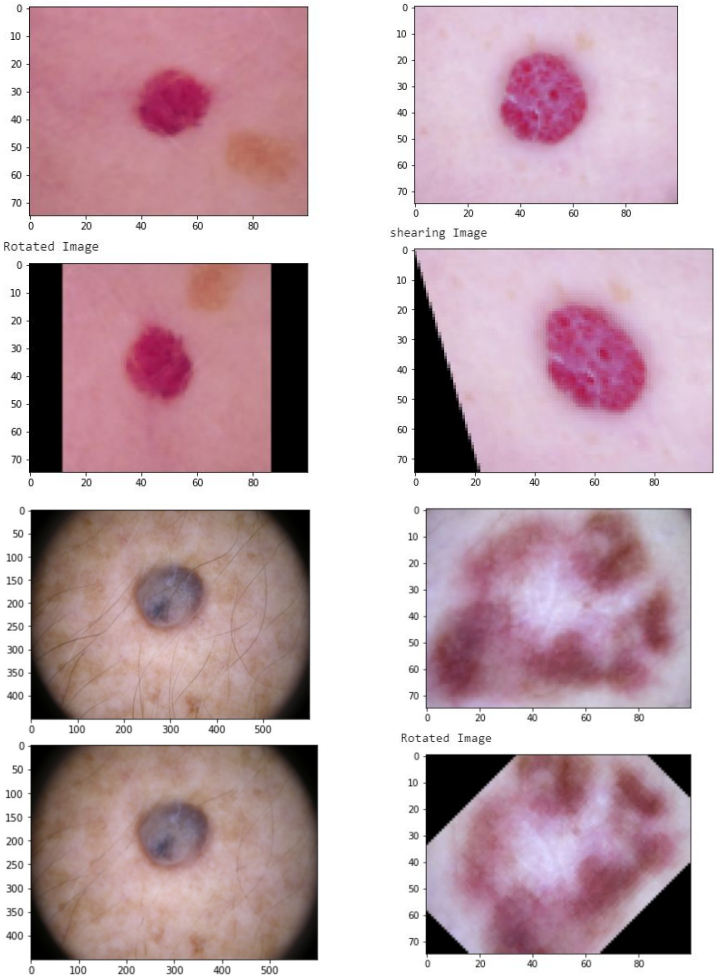
The fewer images, less training samples for the network. Instead of throw away information, we tried to create more.

Data Augmentation

Goal: Create a less unbalanced dataset

By modifying the versions of the images, it is possible to expand the size of the training data.

- Rotating (problem if the lesion is symmetric)
- Shearing
- Blurring (remove hair)

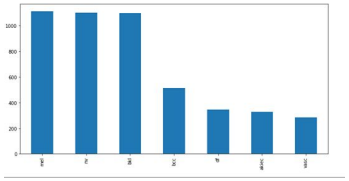


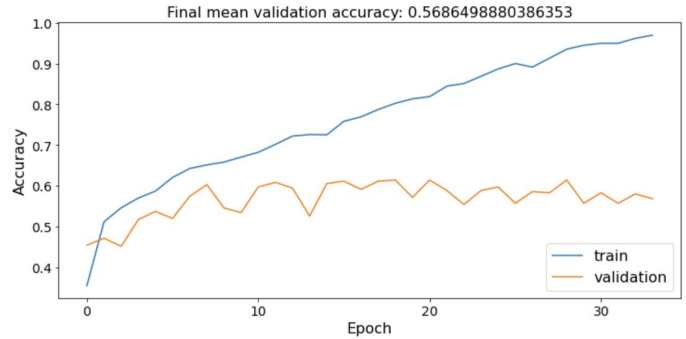
Data Augmentation

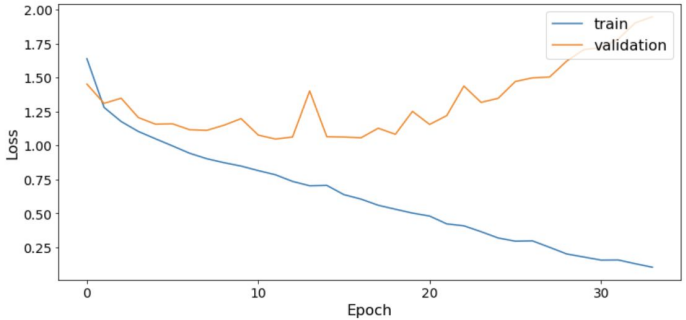
Test accuracy improvement from

57% to **57%**

- so basically no change







Optimizing of the CNN

- We have used the Keras tuner package
  - Optimezies both the number and size of dense lays
- The Hyperband metode
  - Choose many set of hyper-paremeters
  - Evaluates them
  - discards the worst half
  - repeat till only one set is left

Hyperparameters:

|-num_layers: 9
|-tuner/bracket: 3
|-tuner/epochs: 6
|-tuner/initial_epoch: 2
|-tuner/round: 1
|-tuner/trial_id: ef4dfaf2e12c42b22b6bfe627e253459
|-units_0: 64
|-units_1: 10
|-units_2: 200
|-units_3: 124
|-units_4: 42
|-units_5: 184
|-units_6: 236
|-units_7: 62
|-units_8: 88
Epoch 7/17

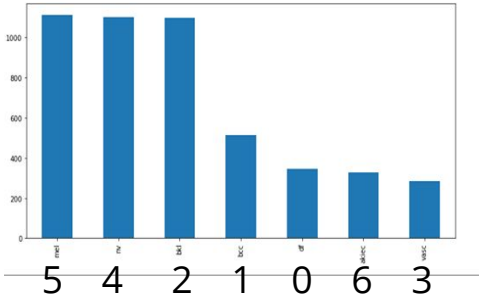
Model: "sequential"

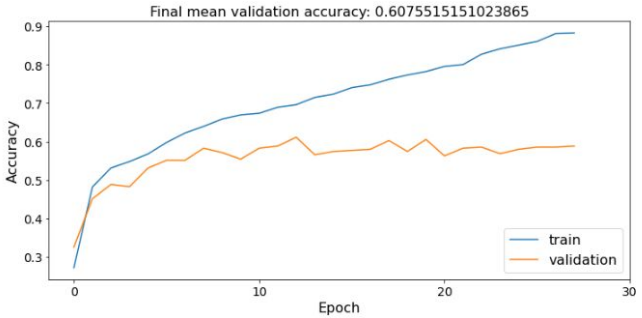
Layer (type)	Output Shape	Param #
conv3d (Conv3D)	(None, 69, 94, 2, 32)	3168
max_pooling3d (MaxPooling3D)	(None, 34, 47, 1, 32)	0
conv3d_1 (Conv3D)	(None, 30, 43, 1, 64)	51264
max_pooling3d_1 (MaxPooling3	(None, 15, 21, 1, 64)	0
conv3d_2 (Conv3D)	(None, 13, 19, 1, 64)	36928
flatten (Flatten)	(None, 15808)	0
dense (Dense)	(None, 216)	3414744
dense_1 (Dense)	(None, 134)	29078
dense_2 (Dense)	(None, 182)	24570
dense_3 (Dense)	(None, 7)	1281

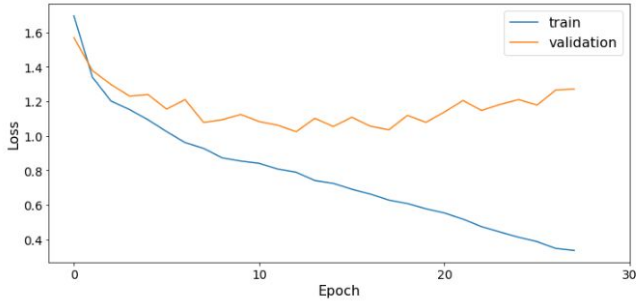
Total params: 3,561,033
Trainable params: 3,561,033
Non-trainable params: 0

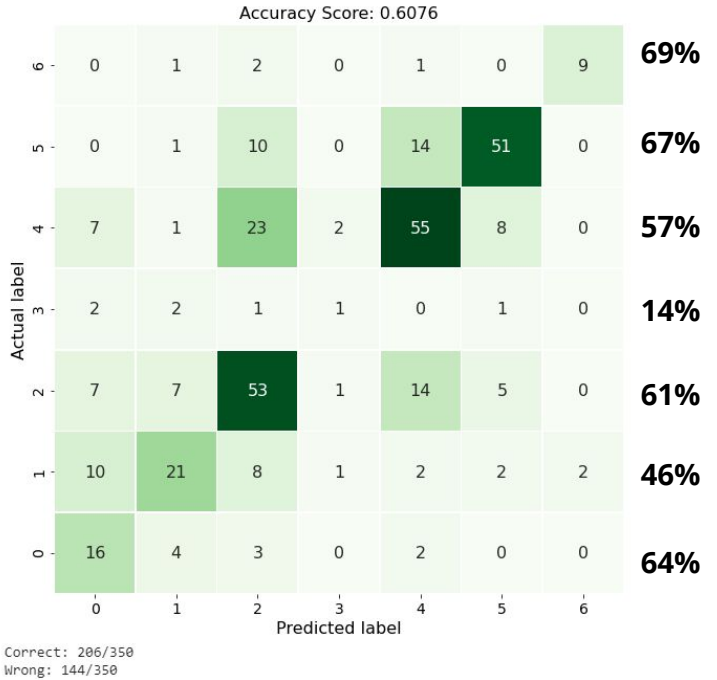
Evaluation of the final result

After optimizing we obtain **61 %** accuracy



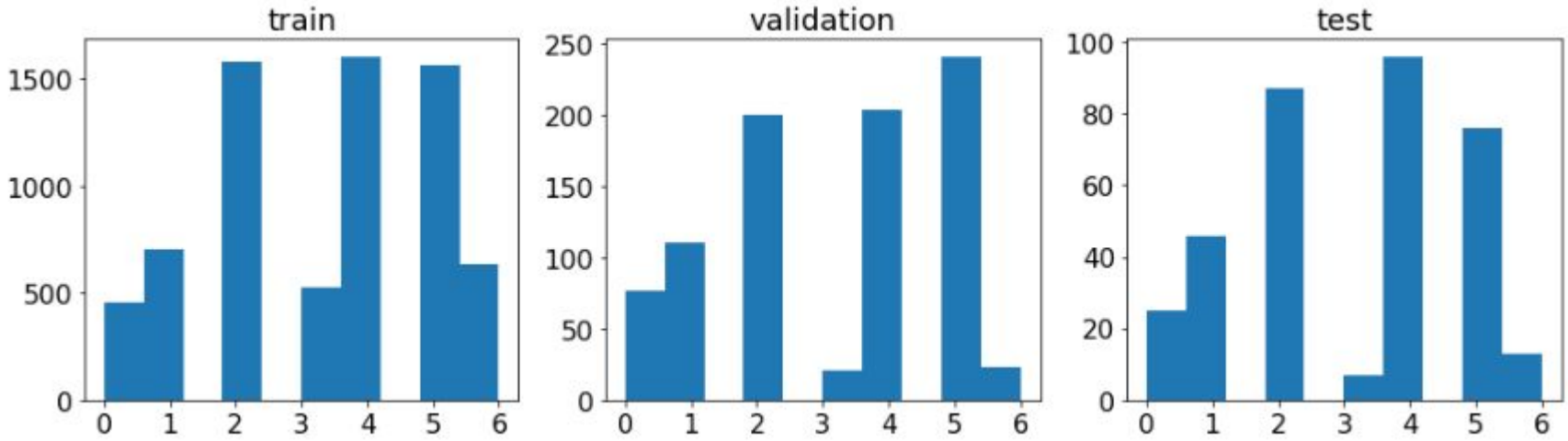




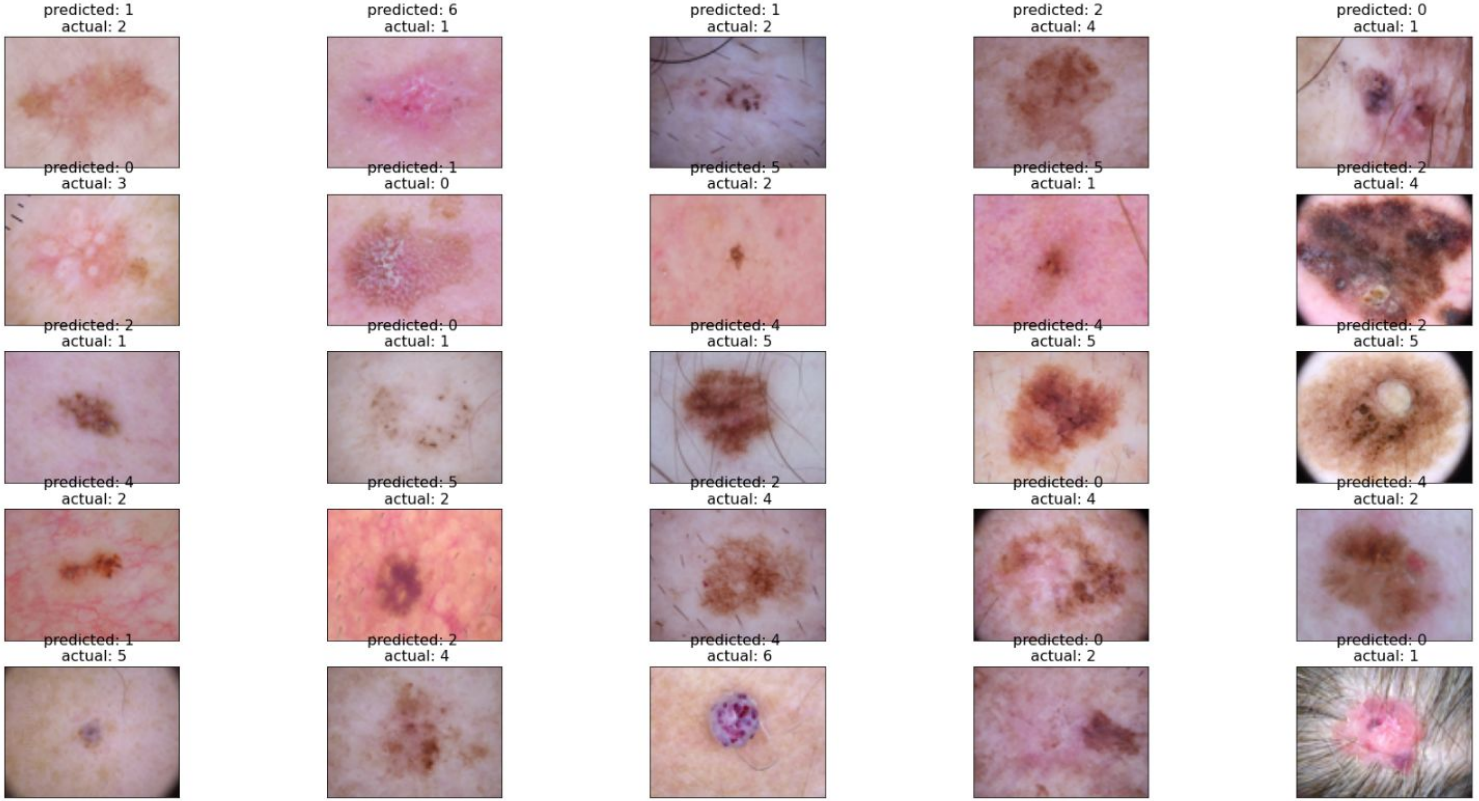


Evaluation of the Result

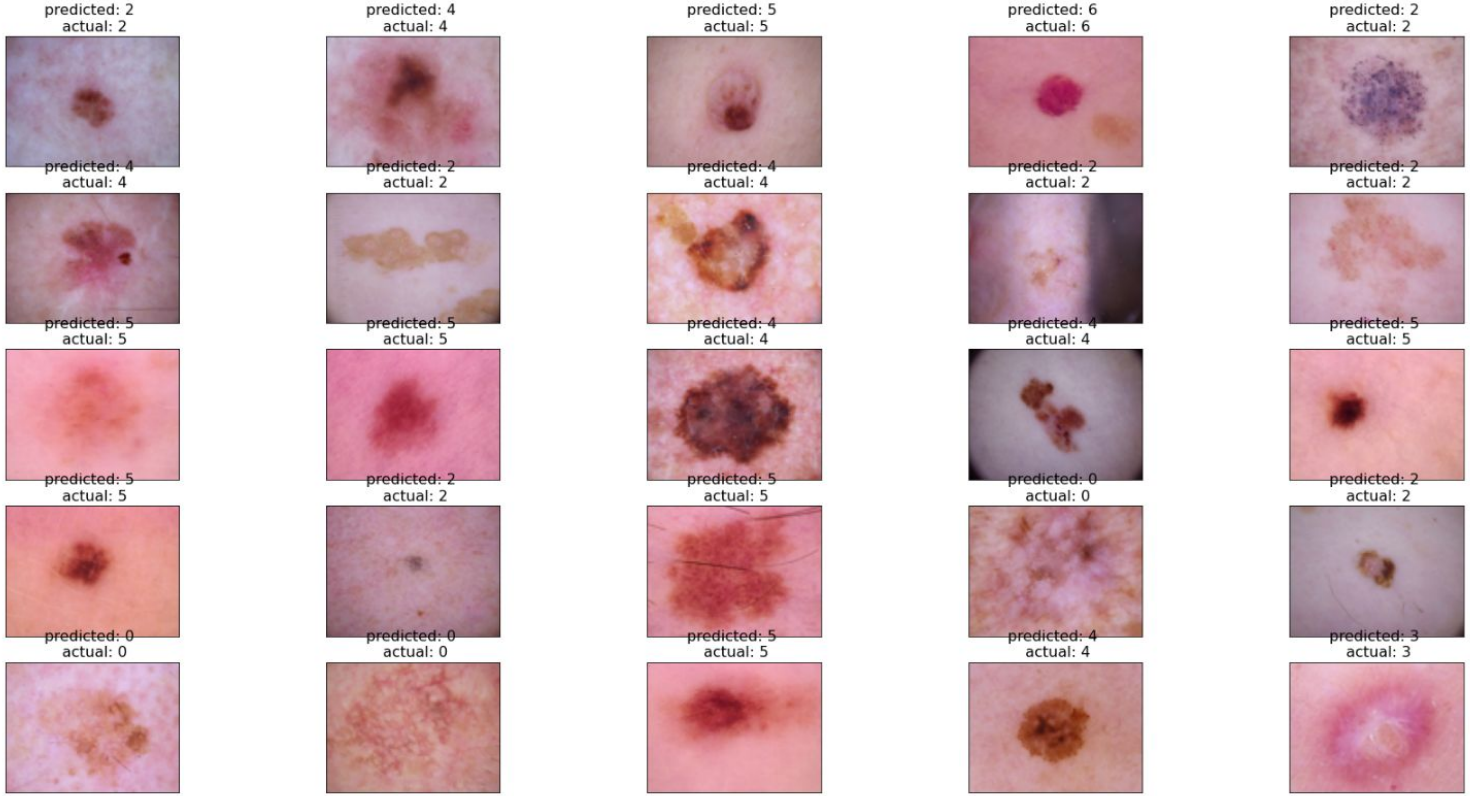
- Among of cell types in the test, train and val



25 random images that the model predict wrong

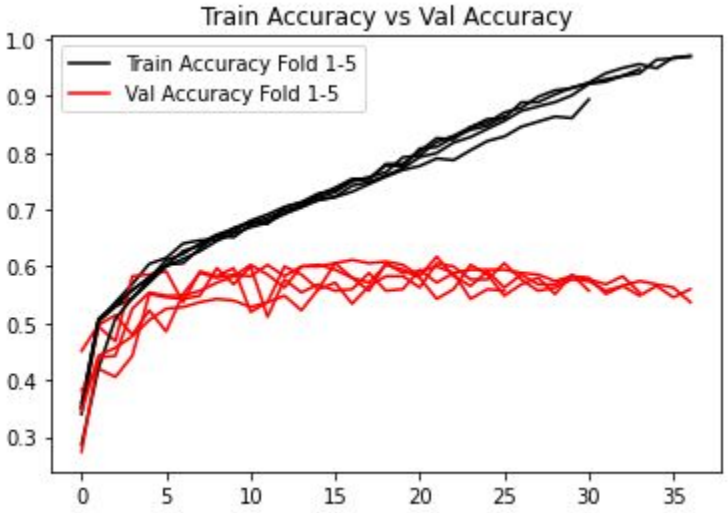


25 random images that the model predict right

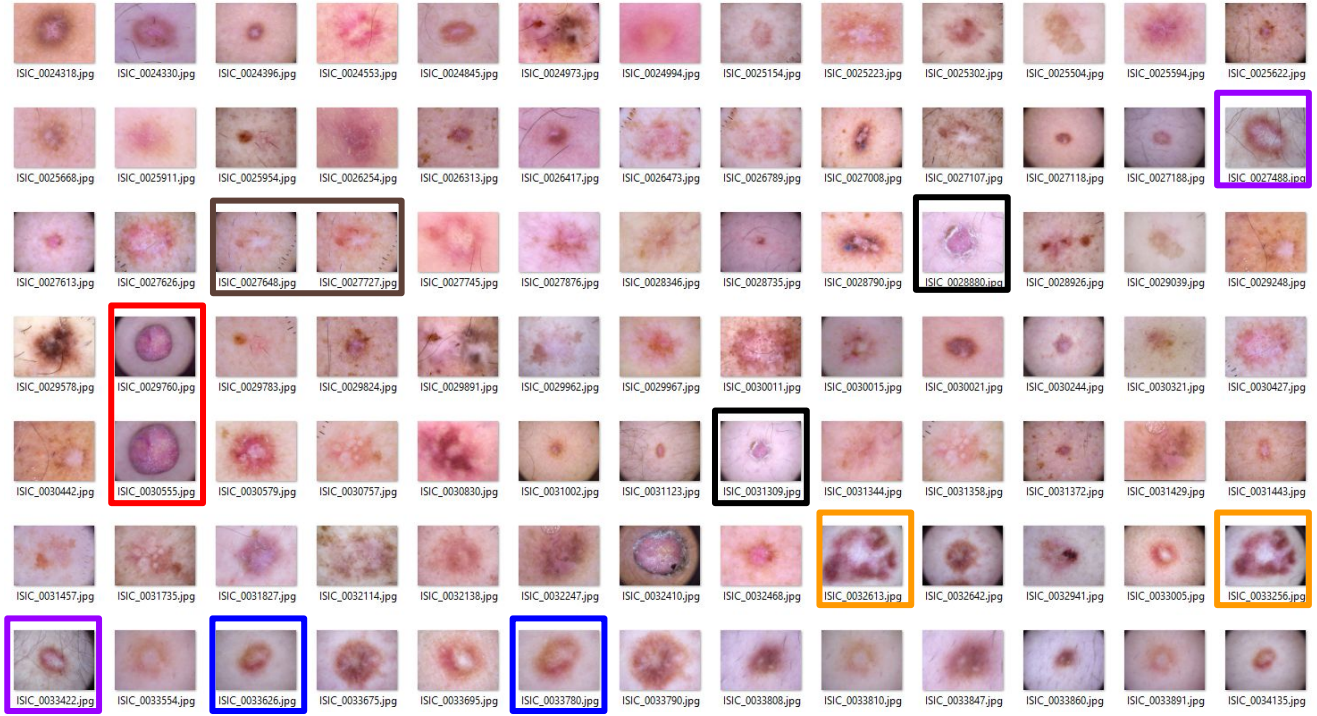


Evaluation of the Result

Cross validation. Showing that the model is robust but overfitted.



Summary of the challenges



Unbalanced data.

The number of images in the datasets, before processing, does not correspond to the number of unique lesions [1].

Old images, the quality is not the best. Collected from the last 20 years [1].

Conclusion

That the data is unbalanced and definitely produces an overfitted network.

Biggest improvements result from including metadata and HP optimization. Lack of information in the data makes it hard to produce good results.

Already few unique images could be reason why data augmentation do not improve the result.

Limited to fair skinned people.

Thanks for listening

References

Link to the officel compentition: https://challenge2018.isic-archive.com/task3/

Dataset: *Tschandl, P. et al. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5:180161 doi: 10.1038/sdata.2018.161 (2018).*

Data: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T

[1] Authors comments on the data: https://www.nature.com/articles/sdata2018161

Tensor flows documentation page: https://www.tensorflow.org/api_docs/python/tf/nn/conv3d

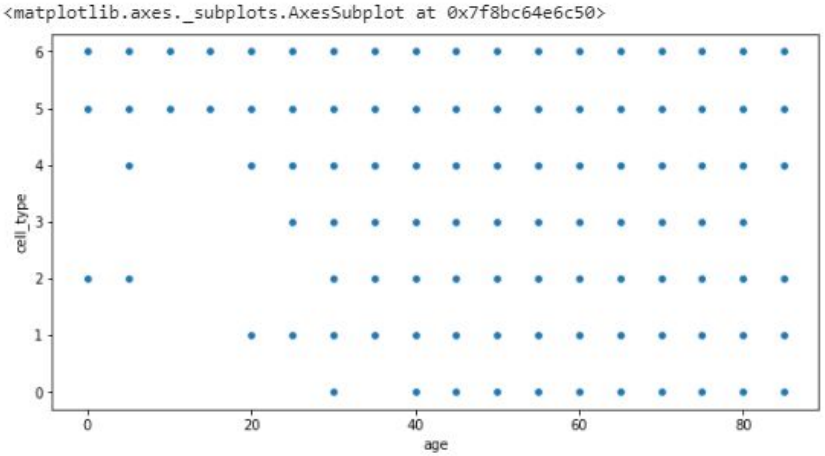
CNN example: *Alexanders Nielsen, Exercise on CNN (CNN_MNISTdata.ipynb). From Week 4, wednesday 13th of May.*

Keras tuner package: https://medium.com/criteo-labs/hyper-parameter-optimization-algorithms-2fe447525903

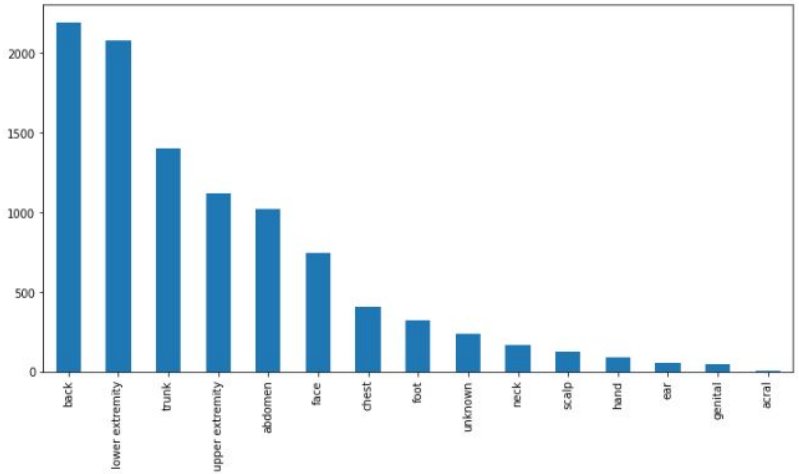
Appendix

Data exploring

Class vs. age

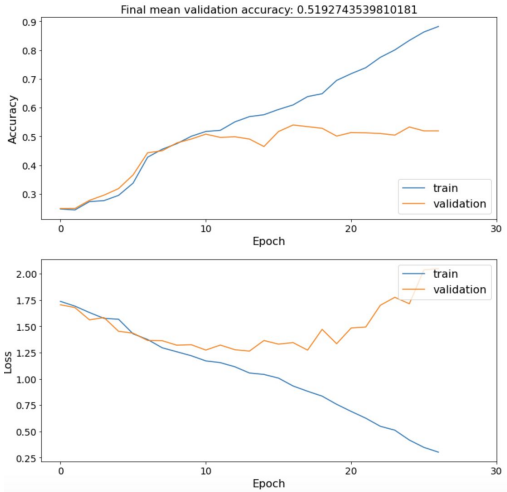


Distribution of the Localization

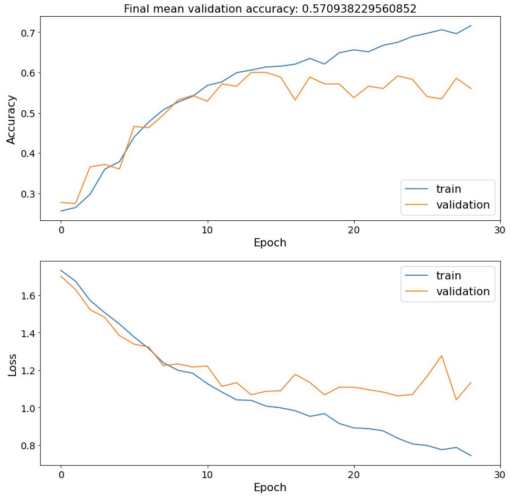


Appendix

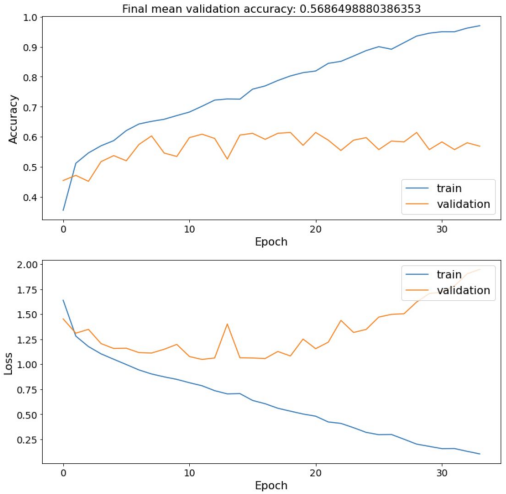
Reducing one (the “nv”) class



Adding metadata



Metadata + data augmentation



Appendix

CNN model Architecture:

- Before HP optimization

Model: "model_3"

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	(None, 75, 100, 3, 1	0	
conv3d_7 (Conv3D)	(None, 71, 96, 2, 64	3264	input_5[0][0]
max_pooling3d_5 (MaxPooling3D)	(None, 35, 48, 1, 64	0	conv3d_7[0][0]
conv3d_8 (Conv3D)	(None, 31, 44, 1, 12	204928	max_pooling3d_5[0][0]
max_pooling3d_6 (MaxPooling3D)	(None, 15, 22, 1, 12	0	conv3d_8[0][0]
conv3d_9 (Conv3D)	(None, 13, 20, 1, 12	147584	max_pooling3d_6[0][0]
flatten_3 (Flatten)	(None, 33280)	0	conv3d_9[0][0]
input_6 (InputLayer)	(None, 3)	0	
concatenate_3 (Concatenate)	(None, 33283)	0	flatten_3[0][0] input_6[0][0]
dense_7 (Dense)	(None, 128)	4260352	concatenate_3[0][0]
dense_8 (Dense)	(None, 128)	16512	dense_7[0][0]
dense_9 (Dense)	(None, 7)	903	dense_8[0][0]

Total params: 4,633,543
Trainable params: 4,633,543
Non-trainable params: 0

- After HP optimization

Model: "sequential"

Layer (type)	Output Shape	Param #
conv3d (Conv3D)	(None, 69, 94, 2, 32)	3168
max_pooling3d (MaxPooling3D)	(None, 34, 47, 1, 32)	0
conv3d_1 (Conv3D)	(None, 30, 43, 1, 64)	51264
max_pooling3d_1 (MaxPooling3	(None, 15, 21, 1, 64)	0
conv3d_2 (Conv3D)	(None, 13, 19, 1, 64)	36928
flatten (Flatten)	(None, 15808)	0
dense (Dense)	(None, 216)	3414744
dense_1 (Dense)	(None, 134)	29078
dense_2 (Dense)	(None, 182)	24570
dense_3 (Dense)	(None, 7)	1281

Total params: 3,561,033
Trainable params: 3,561,033
Non-trainable params: 0

Appendix

What we further did to optimize:

- chance the learning rate
- Split the train, test and val different
- more epoch
- smaller group with equal size
- change the seed (random state)