Improving the existing face age recognition solution

Course: Podstawy rozpoznawania obrazów

Author: Tomasz Główka

Table of contents

Introduction

The main purpose of the project is to improve the accuracy of the existing solution that predicts the age of a person from a given image and propose a production environment testing pipeline to measure face age recognition capabilities of the improved solution.

Existing solution

The existing solution used three types of network CNN architecture:

- SimpleConvNet 100x100,

- SimpleConvNet 224x224,

- EffictientNet 224x224.

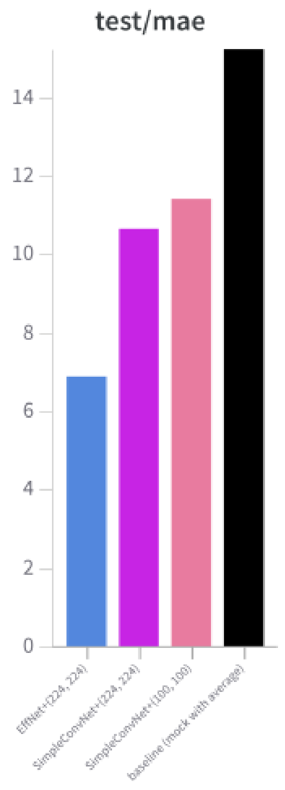


Figure 1: Mean absolute error (MAE) on test set generated by weights which obtained best validation loss on each run. EfficientNet scored the best with 6.896MAE.

As we can see from the results the EfficientNet architecture score was the greatest = 6.896. The "best-checkpoint.ckpt" from the existing solution was tested on the test dataset (generated to test the improved solution) and it scored 6.3050. The goal is to achieve better result.

Improved solution

The improved solution was implemented from scratch. However, the general approach was imported from the existing solution i.e.:

- Training/Validation datasets generation,
- Test dataset generation,
- Age clipping (age > 80 is clipped to the 80),

- SimpleConvNet 224x224 architecture,
- EfficientNet 224x224 architecture.

To improve the existing solution the following ideas were tested:

- K-fold validation (K = 10),
- Removing the RandomHorizontalFlip when reading the image,
- Adam optimizer learning rate modifications.

K-fold validation

K-fold validation is a technique that splits the training dataset into K folds. 1 fold is used to validate the model whereas K-1 folds are used to train the model. The K-fold technique was implemented for the K=10. The training and validation sets were merged together. The full training consisted of the 10 phases, 1 phase for each fold. Each phase ran for 10 epochs and produced 1 best checkpoint according to the best validation loss.

Removing the RandomHorizontalFlip

It was decided to remove RandomHorizontalFlip from the training, validation and testing parts because it (unnecessarily) modifies original data. Similar image transforms were used to create production environment test pipeline.

Adam optimizer learning rate modifications

Adam optimizer learning rate modifications were introduced to see how it affects training part and final accuracy of the model. Learning rate values used for the experiment were: 0.01, 0.001, 0.0001.

Experiments

To find a model with the best accuracy few experiments were conducted. The experiments were conducted for the K-fold validation (K=10), learning rate values: 0.01, 0.001, 0.0001 and for the “SimpleConvNet 224x224” and “EfficientNet 224x224” architectures.

Learning process

For each combination of the architectures and learning rate values training process was run. The results are depicted below:

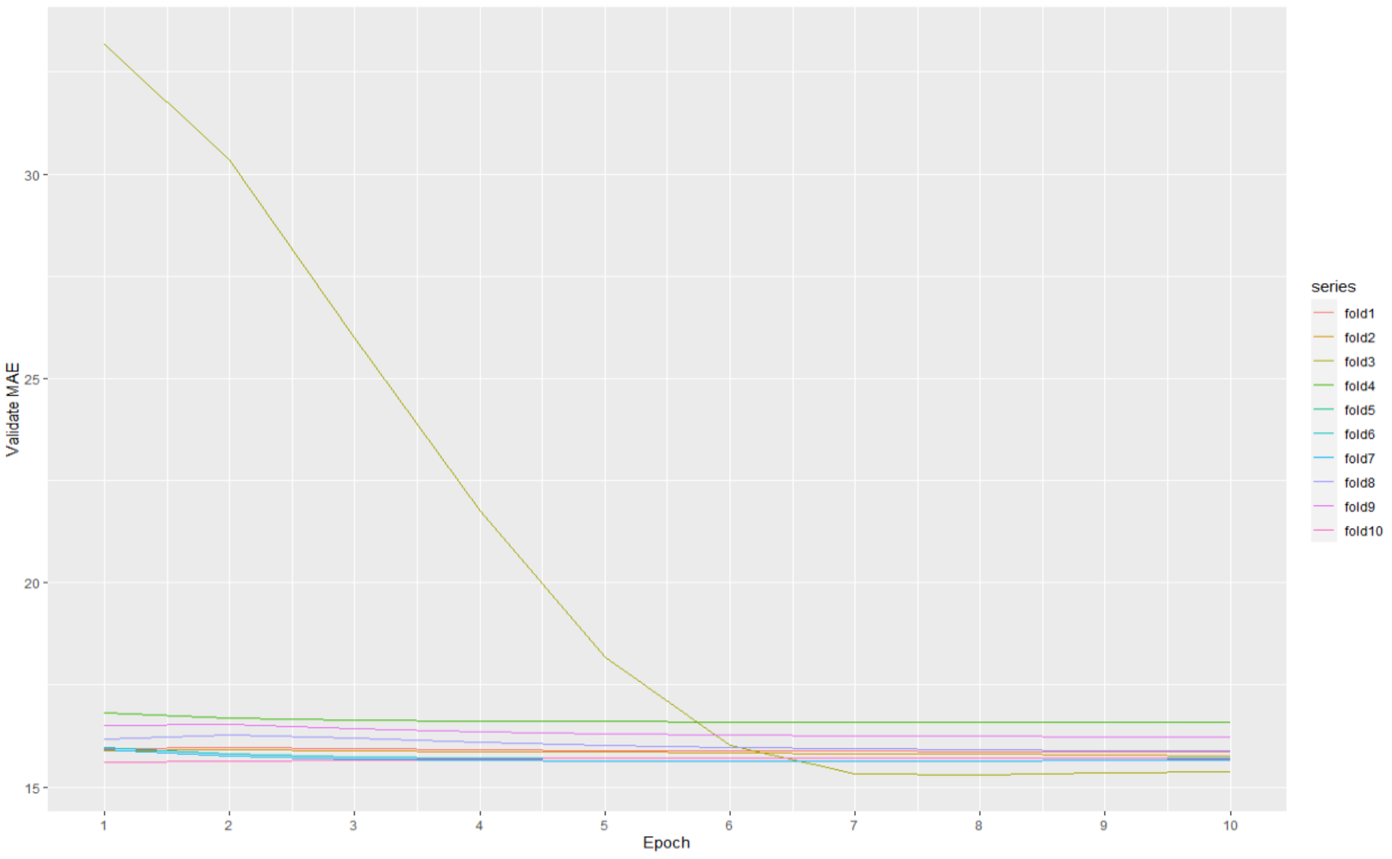


Figure 2: SimpleConvNet 224x224, LR = 0.01.

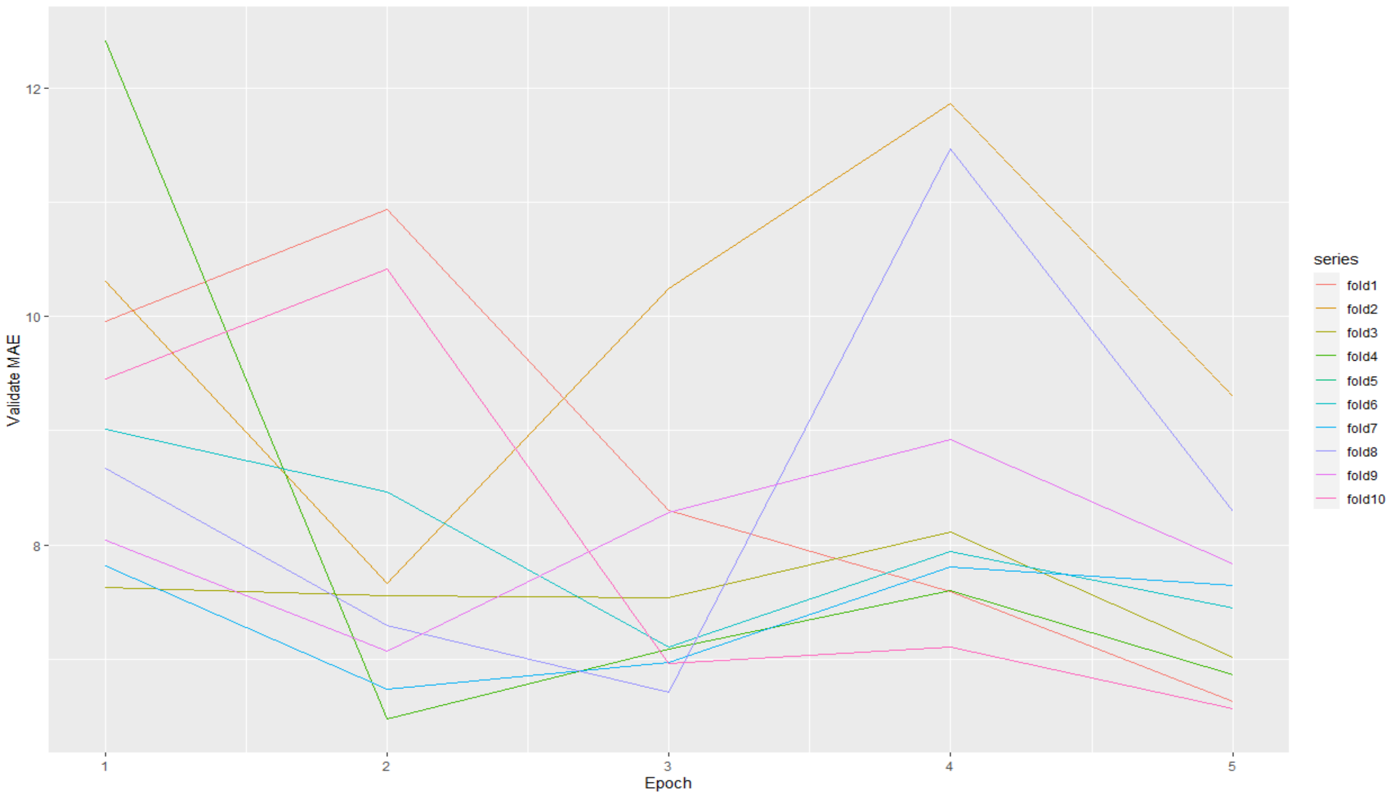


Figure 3: EfficientNet 224x224, LR = 0.01.

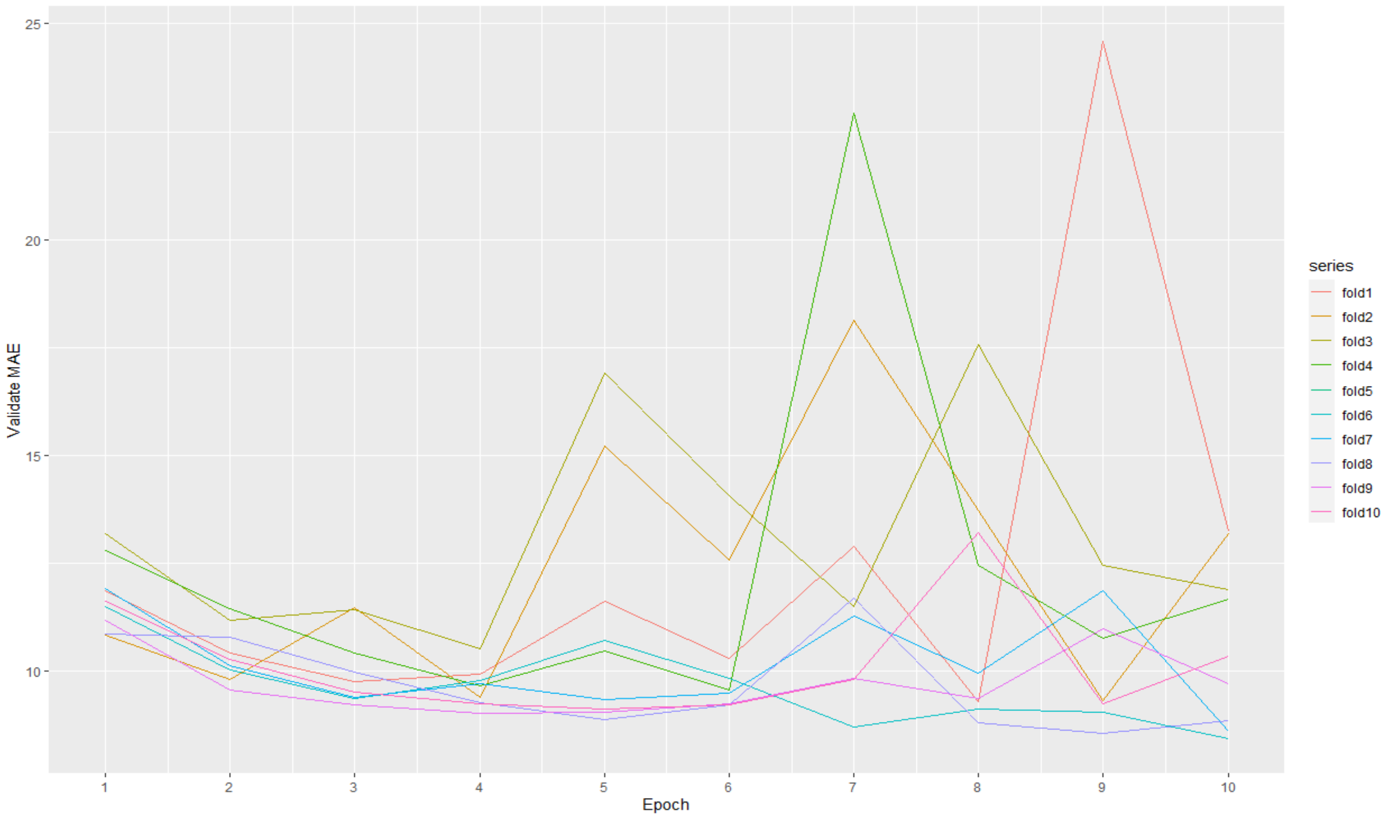


Figure 4: SimpleConvNet 224x224, LR = 0.001.

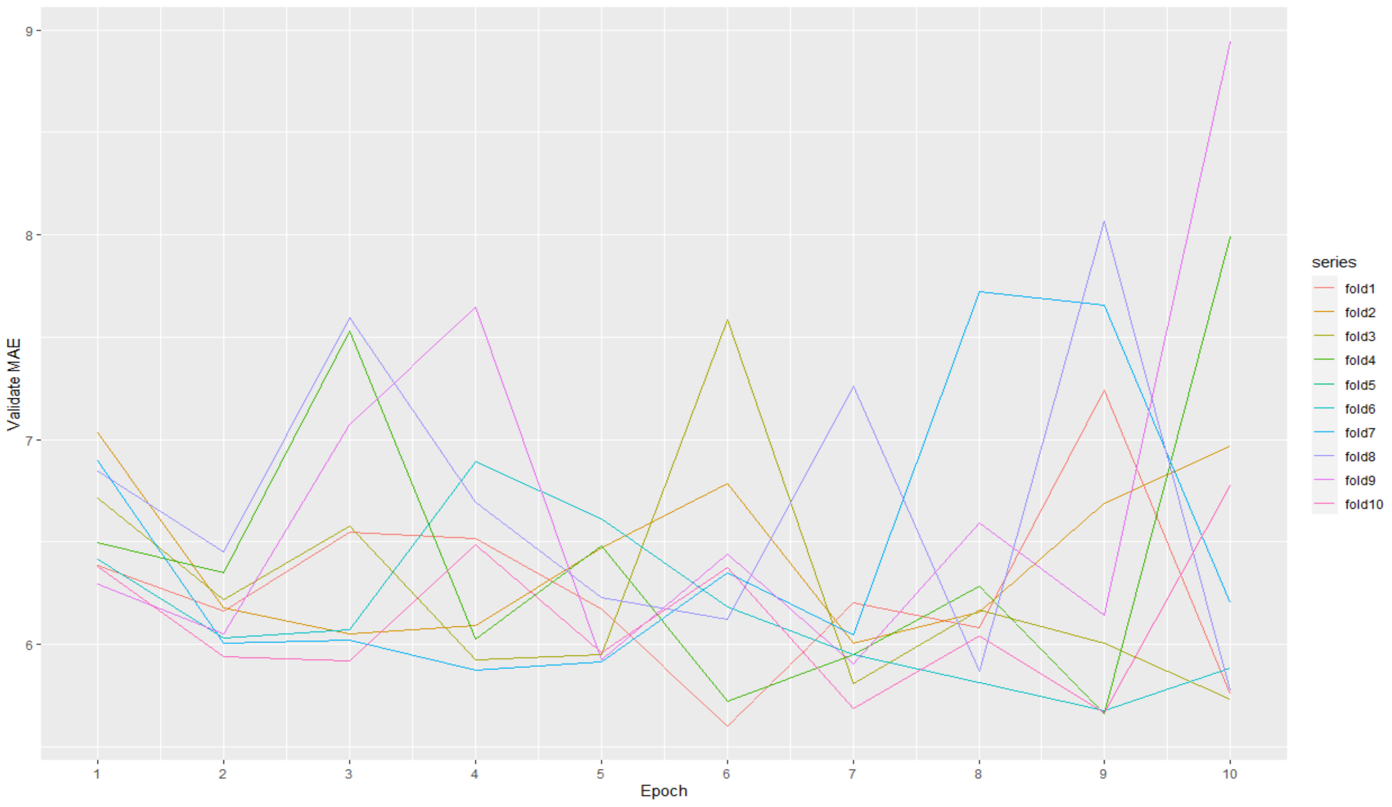


Figure 5: EfficientNet 224x224, LR = 0.001.

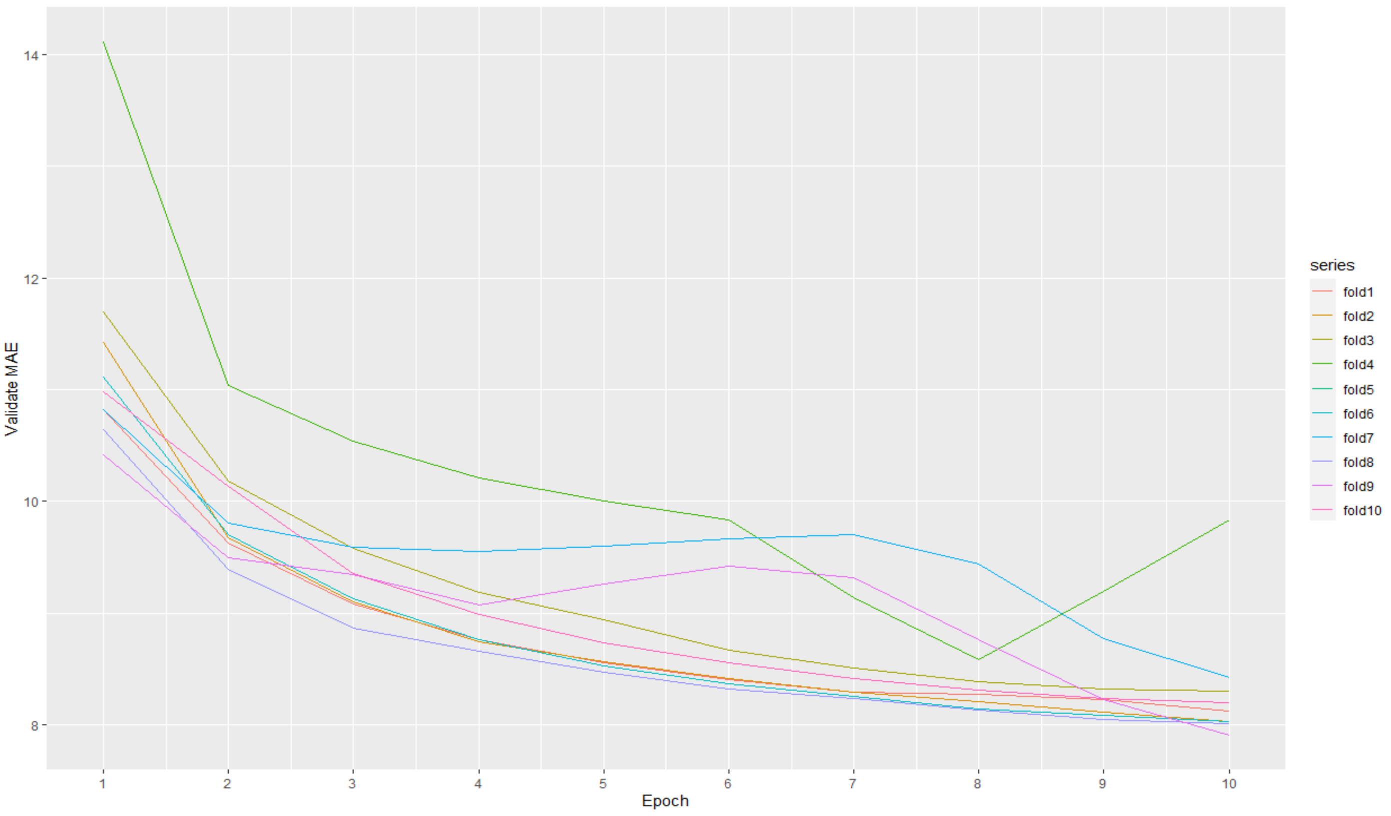


Figure 6: SimpleConvNet 224x224, LR = 0.0001.

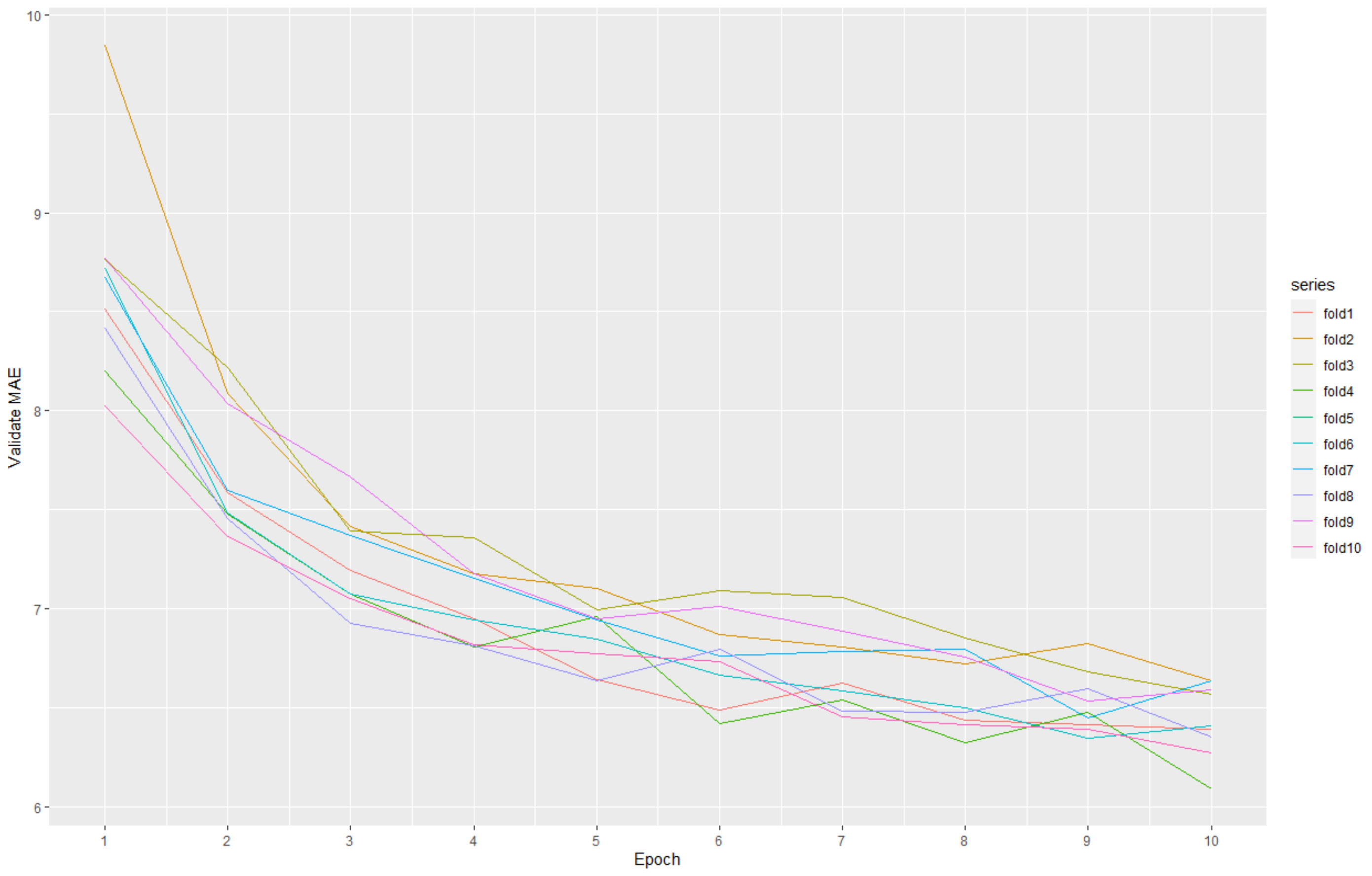


Figure 7: EfficientNet 224x224, LR = 0.0001.

The summary of the training process for each combination is presented in the table below:

	Fold no.	1	2	3	4	5	6	7	8	9	10
Val MAE min	SimpleConvNet 224x224, LR = 0.01	15.86	15.77	15.30	16.58	15.70	15.70	15.63	15.89	16.23	15.61
	EfficientNet 224x224, LR = 0.01	6.63	7.66	7.02	6.48	7.10	7.10	6.74	6.72	7.07	6.57
	SimpleConvNet 224x224, LR = 0.001	9.30	9.32	10.53	9.57	8.45	8.45	8.61	8.57	9.04	9.12
	EfficientNet 224x224, LR = 0.001	5.60	6.00	5.73	5.66	5.68	5.68	5.88	5.77	5.91	5.67
	SimpleConvNet 224x224, LR = 0.0001	8.13	8.03	8.31	8.59	8.03	8.03	8.42	8.01	7.91	8.20
	EfficientNet 224x224, LR = 0.0001	6.39	6.64	6.57	6.09	6.35	6.35	6.45	6.35	6.54	6.27

Figure 8: Val MAE min values for each fold no. in training process.

As we can see from the table the lower the learning rate value the better results were achieved for the SimpleConvNet architecture. However, the same does not apply for the EfficientNet architecture. The best results for the EfficientNet architecture were achieved for the learning rate = 0.001.

Test validation

All 6 experiments were tested on the test dataset generated for the sake of this project. Each experiment produced 10 best checkpoints based on the best validation loss (for each fold run). To get a test score of a single experiment each checkpoint was loaded to make its own prediction. The test score of the experiment was the Mean Absolute Error (MAE) of the mean value of the 10 predictions. The test results are depicted in the table below:

Experiment	**Test score**
SimpleConvNet 224x224, LR=0.01	19.7310
EffNet 224x224, LR=0.01	7.0518
SimpleConvNet 224x224, LR=0.001	9.9090
EffNet 224x224, LR=0.001	5.8364
SimpleConvNet 224x224, LR=0.0001	9.6467
EffNet 224x224, LR=0.0001	6.5083
Existing	6.3050

Figure 9: Test score for each experiment.

As we can see the EfficientNet scored better than the SimpleConvNet each and every time. Learning rate modifications had significant impact regarding obtained test score. The best score was obtained by the "EfficientNet 224x224, LR=0.001" = 5.8364. The proposed model is better than the existing one that scored the 6.3050 on the same test dataset.

Production environment pipeline

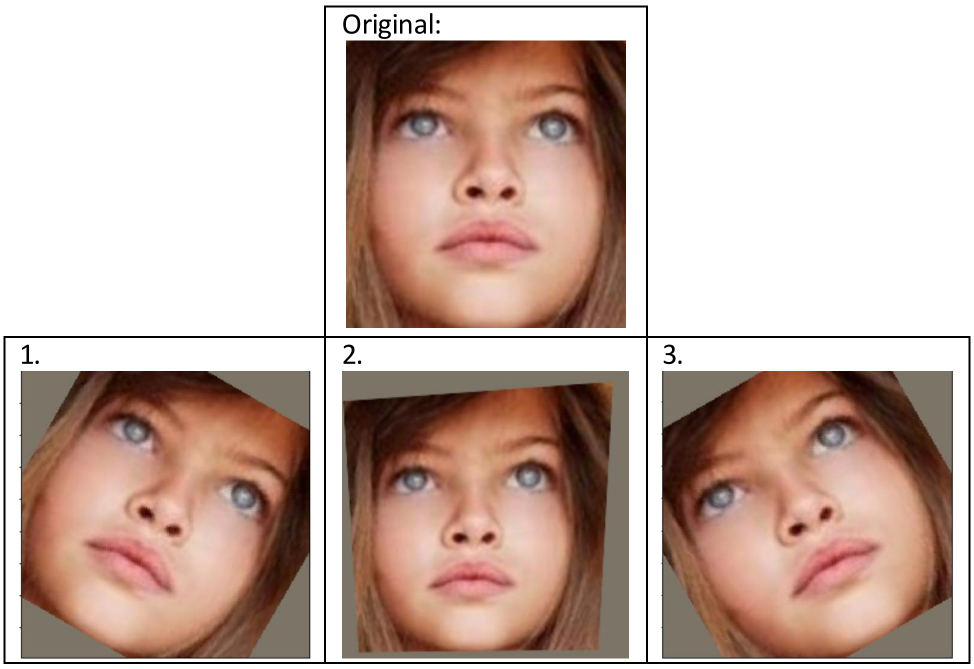
To measure the accuracy of the proposed solution additional production environment pipeline was created. The main purpose of the pipeline was to test the model on the face images similar to the ones from the production environment, i.e. transformed images. To mimic the production environment the "transforms" module from the "torchvision" module was used. The pipeline consisted of 8 transforms:

1. RandomPerspective,
2. RandomRotation,
3. RandomAffine,
4. ElasticTransform,
5. ColorJitter,
6. GaussianBlur,
7. RandomAdjustSharpness,
8. RandomHorizontalFlip.

The 8 transforms were used with different set of parameters and eventually the 12 transforms were created ("FaceAgeTransforms" class , "create_transforms" method):

1. random_rotation(degrees=(-30, -30)),
2. random_perspective(distortion_scale=0.25),
3. random_rotation(degrees=(30, 30)),
4. random_affine(scale=(0.75, 0.75)),
5. random_affine(scale=(1.25, 1.25)),
6. elastic_transform(alpha=100.0, sigma=5.0),
7. color_jiter(brightness=(1.5, 1.5)),
8. color_jiter(contrast=(1.5, 1.5)),
9. color_jiter(saturation=(1.5, 1.5)),
10. color_jiter(hue=(0.3, 0.3)),
11. gaussian_blur(kernel_size=(5, 9), sigma=5),
12. random_horizontal_flip().

Each of the transforms is depicted below:



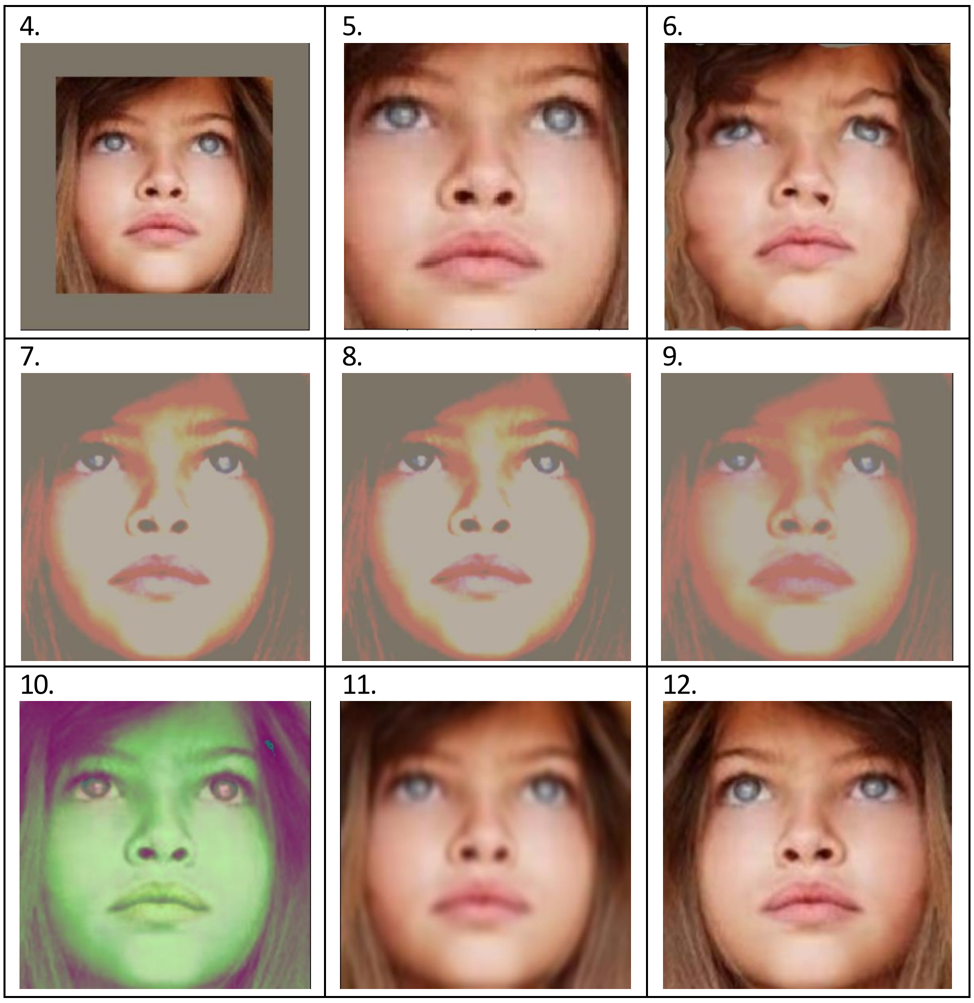


Figure 10: Test pipeline transforms examples.

Transforms test

Each of the experiments (as well as the existing solution) were tested on the pipeline to see what is the score for the experiment when images were transformed. Test pipeline results are presented in the table below:

		Experiment						
		Simple, LR=0.01	Eff, LR=0.01	Simple, LR=0.001	Eff, LR=0.001	Simple, LR=0.0001	Eff, LR=0.0001	Existing
Transform	Without	19.7310	7.0518	9.9090	5.8364	9.6467	6.5083	6.3050
	1.	19.7333	12.2790	21.0475	9.4413	21.6735	10.9004	12.8795
	2.	19.7338	9.0071	13.6651	7.5723	12.5589	9.0263	7.8590
	3.	19.7326	12.5944	21.9256	9.0247	22.0728	11.2523	13.1291
	4.	19.7337	12.2127	15.4832	11.0284	14.9404	12.2077	9.9430
	5.	19.7309	9.1359	14.3158	6.8700	13.8908	9.6520	8.4072
	6.	19.7307	7.9257	10.2534	6.8592	9.9106	9.8891	7.1105
	7.	19.7326	11.7167	16.0142	11.6804	14.3361	16.9145	11.5674
	8.	19.7327	11.3139	15.8524	11.0954	14.0649	16.4037	11.2953
	9.	19.7328	10.7374	16.1571	10.2429	14.4881	15.3642	10.2976
	10.	19.7326	8.1595	15.6020	7.2431	14.0732	9.8562	7.6202
	11.	19.7296	8.8467	11.3151	6.9767	11.0115	12.3576	6.8624
	12.	19.7328	7.2589	10.5549	5.9382	10.1685	6.6396	6.4190

Figure 11: Transform scores for each experiment.

Interestingly, “EfficientNet 224x224, LR=0.001” wasn’t better each and every time in the test pipeline. Transforms 4, 7 and 11 turned out to be slightly better for the existing solution. However, rest of the transforms were better for the “EfficientNet 224x224, LR=0.001” so the proposed solution is still considered to be better than the existing solution. The picture below presents the score for each transform (dots) and the baseline (straight line) of the existing solution and the improved “EfficientNet 224x224, LR=0.001” architecture:

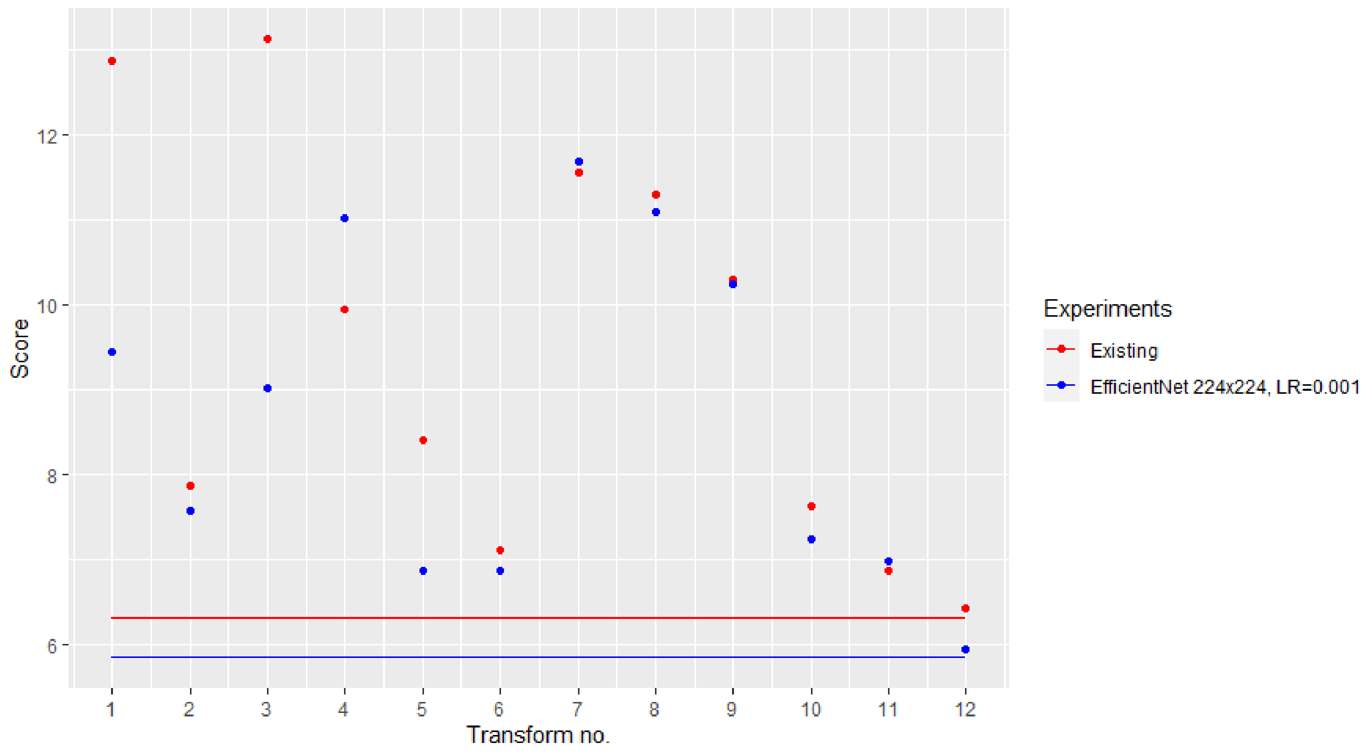


Figure 12: Scores obtained for each transform and the baselines.

Further improvements

The proposed solution turned out to be better at predicting face age based on a given image. However, obtained score is far from the ideal and it is believed that there is a place for improvement. For further improvements the following ideas might be good point area for research:

- Apply train data augmentation by using different transforms.
- Increase the number of epochs in training phase.
- Try different K values regarding K-fold validation technique.
- Try different cross-validation techniques.
- Try different learning rate values (e.g. 0.0005).