Aestheticae estimation

Igor Ratajczyk, Eng
igor@ratajczyk.eu

Aesthetica? Aestheticae?

One may say there is just one aesthetica. Such approach would be called Platonic.

Another one may argue, that there is plethora of equivalent aesthetics.

Aestheticae in photography

Aestheticae in photography

Please check bartekpog@Instagram

Problem Statement

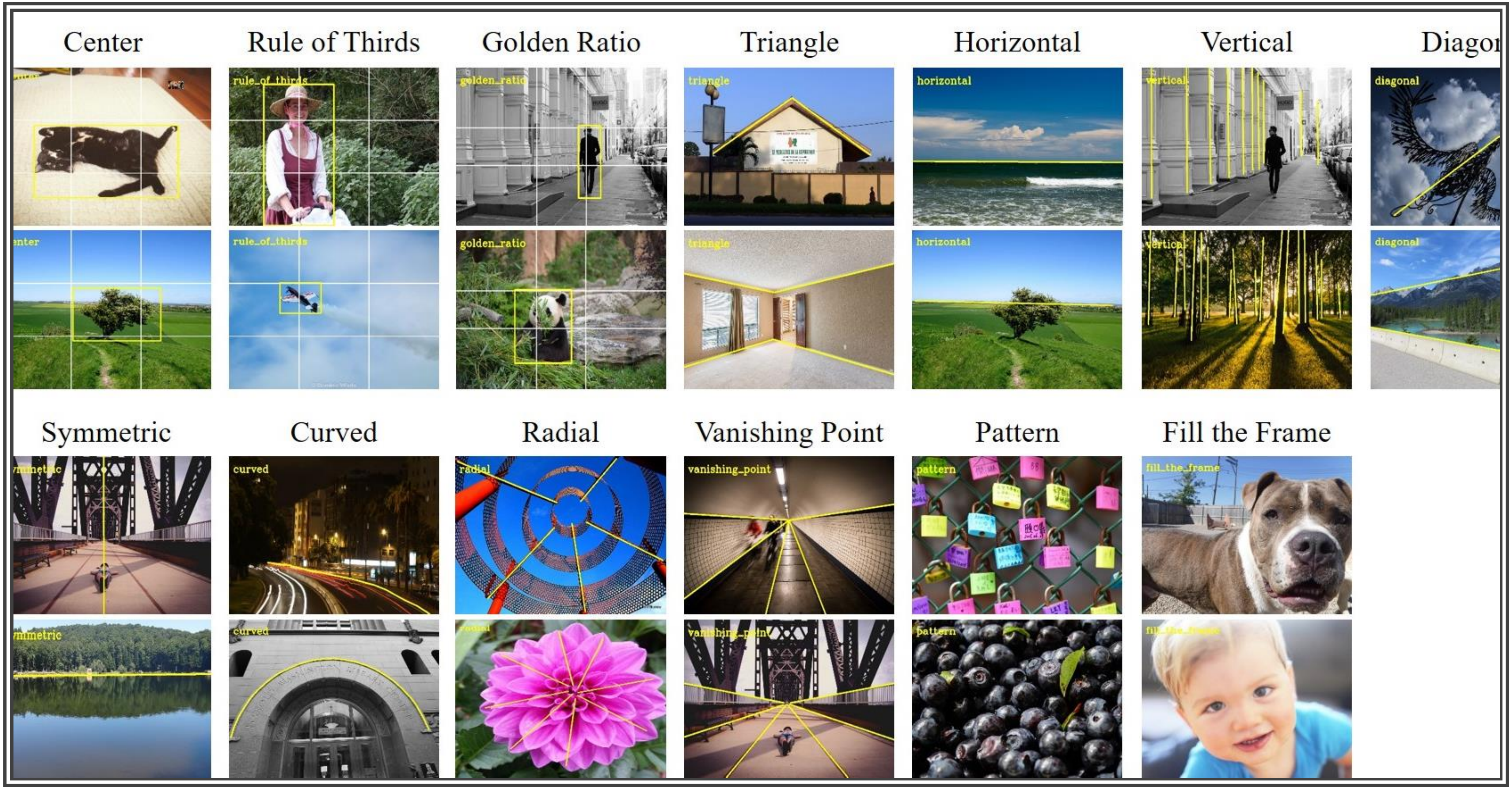
Let us consider problem of aesthetics of photographies. Images do cover various though limited topics:

- Animals
- Plant
- Human
- Static
- Architecture
- Landscape
- Cityscape
- Indoor
- Night

Dataset

Image Composition Assessment Dataset will be used (9.5k images):

https://github.com/bcmi/Image-Composition-Assessment-Dataset-CADB


Center
Rule of Thirds
Golden Ratio
Triangle
Horizontal
Vertical
Diago
Symmetric
Curved
Radial
Vanishing Point
Pattern
Fill the Frame

Custom loss function

As aesthetics is barely objective, it is required to emphasize truly meaningful samples:

$$J = \frac{1}{N}\sum_{i=1}^{N}\frac{(\hat{y}_i - y_i)^2}{\sigma_i + \epsilon}$$

$$\epsilon = 0.02$$

Epsilon has been introduced to maintain loss function J finite.

TensorFlow

To sum up, the project differs from default image processing as:

- Regression is performed (instead of classification)
- Data has sample-wise-weights
- Barely any augmentation can be performer

Sadly, TF do not suport such complex task in its default interface, especially no tf.data.Dataset can be easily creadted (for .cache() & .prefetch() usages).

Dataset

- 1600/10000 Images for training
- 400/10000 Images for validation
- Images resized to (64,64) without padding

Base model

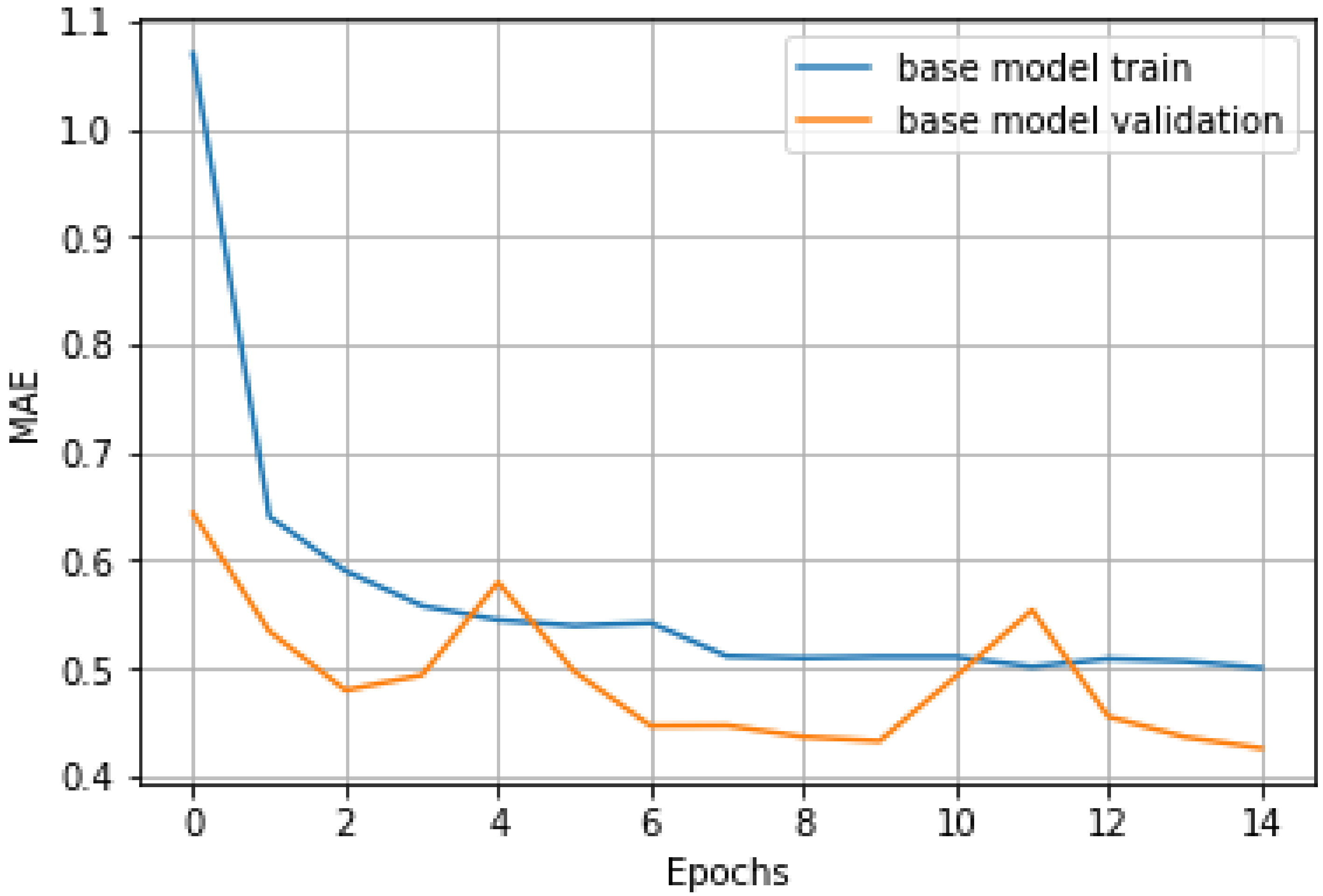
```
model: tf.keras.models.Model = keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='linear'),
])

model.compile(
    optimizer='adam',
    loss = tf.keras.losses.MeanSquaredError(),
    metrics=[tf.keras.losses.MeanSquaredError(), tf.keras.losses.MeanAbsoluteError()]
)
```

Base model performance

MAE has been chosen for its relative ease of result interpretation.

Approximately MAE at 0.5 is required to perform discrete classification of aesthetics properly.

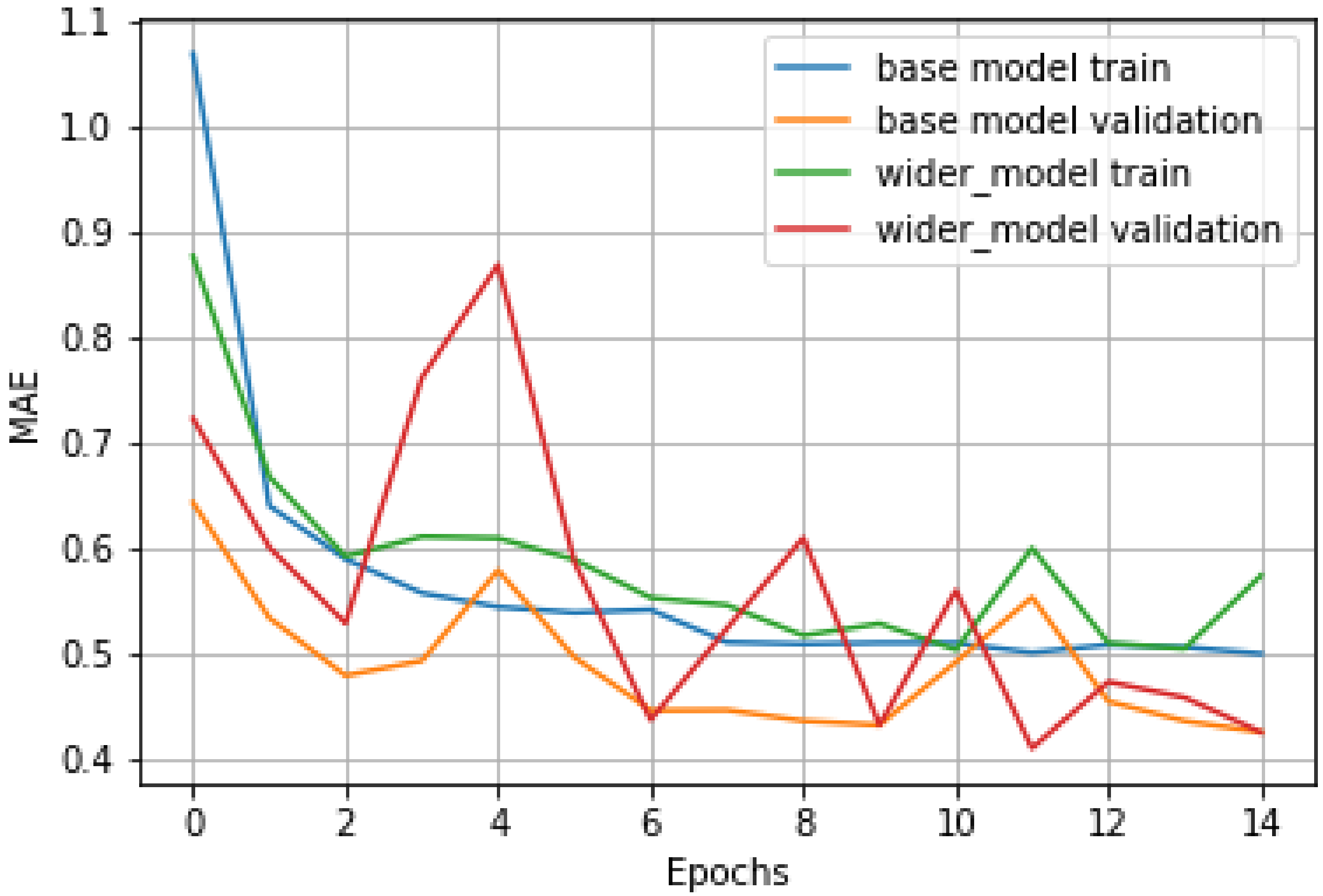


Wider model

```
model_wider: tf.keras.models.Model = keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(5, 5), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(7, 7), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='linear'),
])
```

Wider model performance

Training proces is much more chaotic on default learning rate

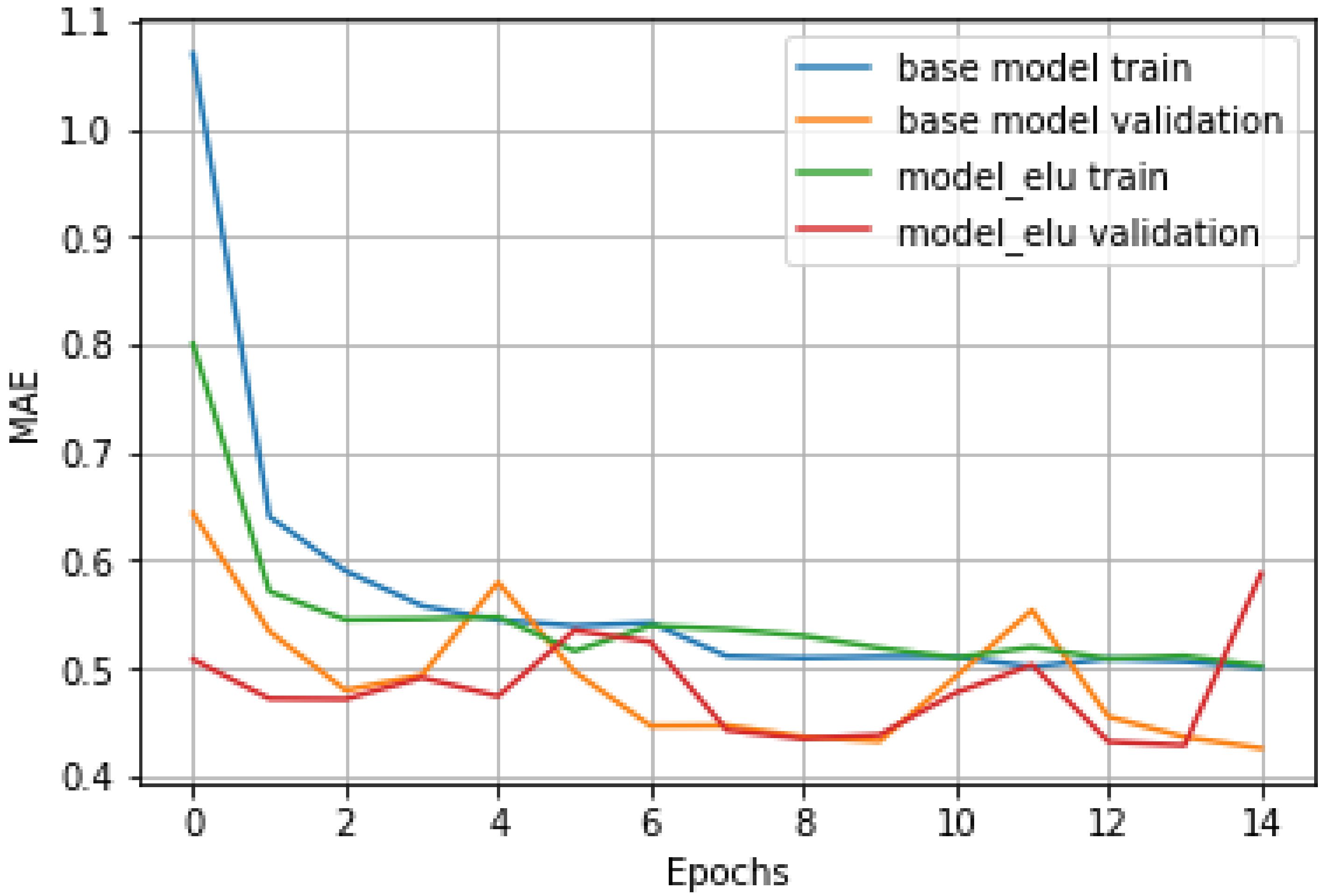


ELU model

```
model_elu: tf.keras.models.Model = keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='elu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='elu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='elu'),
    tf.keras.layers.Dense(64, activation='elu'),
    tf.keras.layers.Dense(1, activation='linear'),
])
```

ELU model performance

ELU model is more chaotic than ReLU base model.

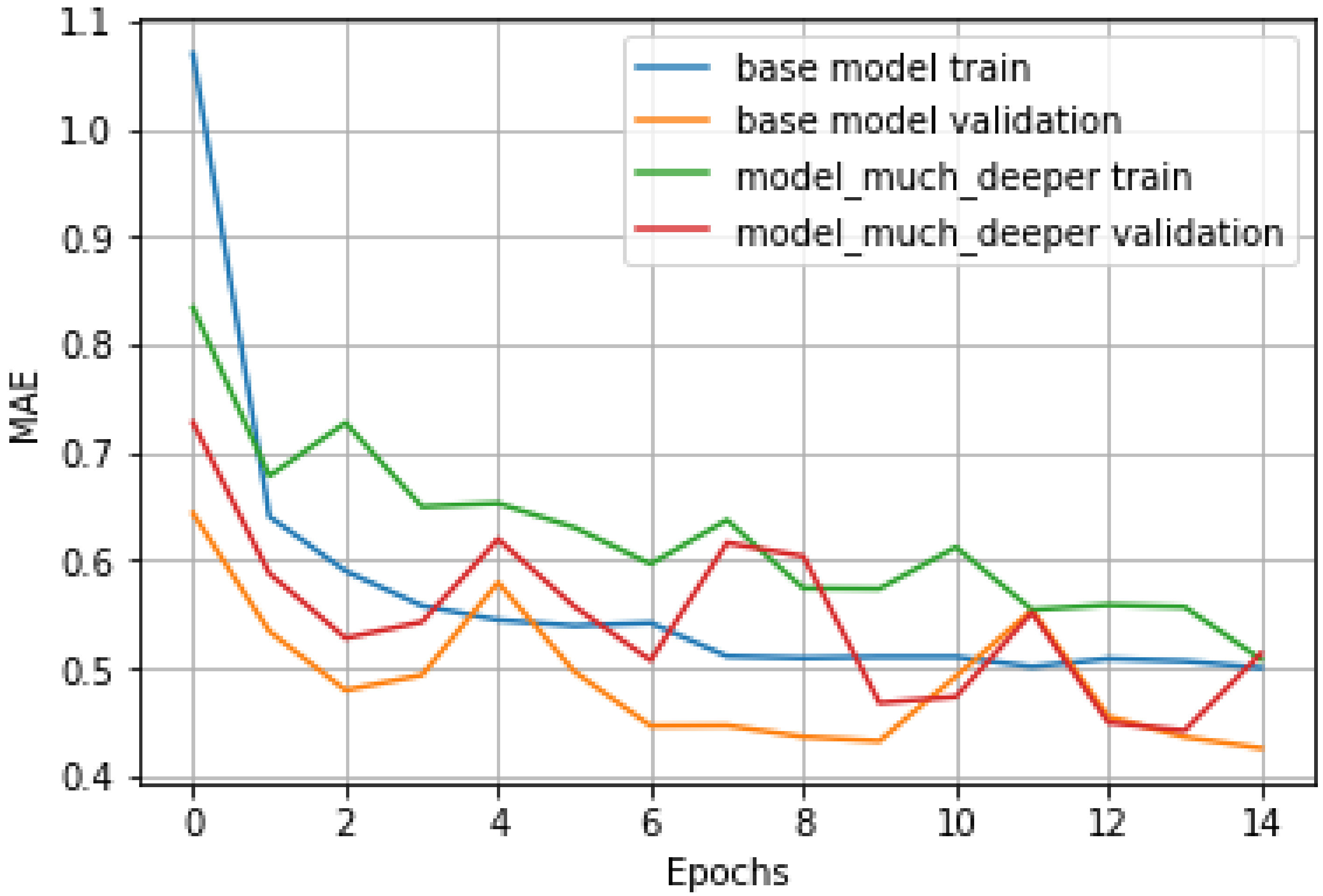


Very Wide and Deep Model

```
vw_model: tf.keras.models.Model = tf.keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(11, 11), activation='relu'),
    tf.keras.layers.MaxPool2D((5,5)),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='linear'),
])
```

VW model performance

- Learning proces is slower and less stalbe than basic one.

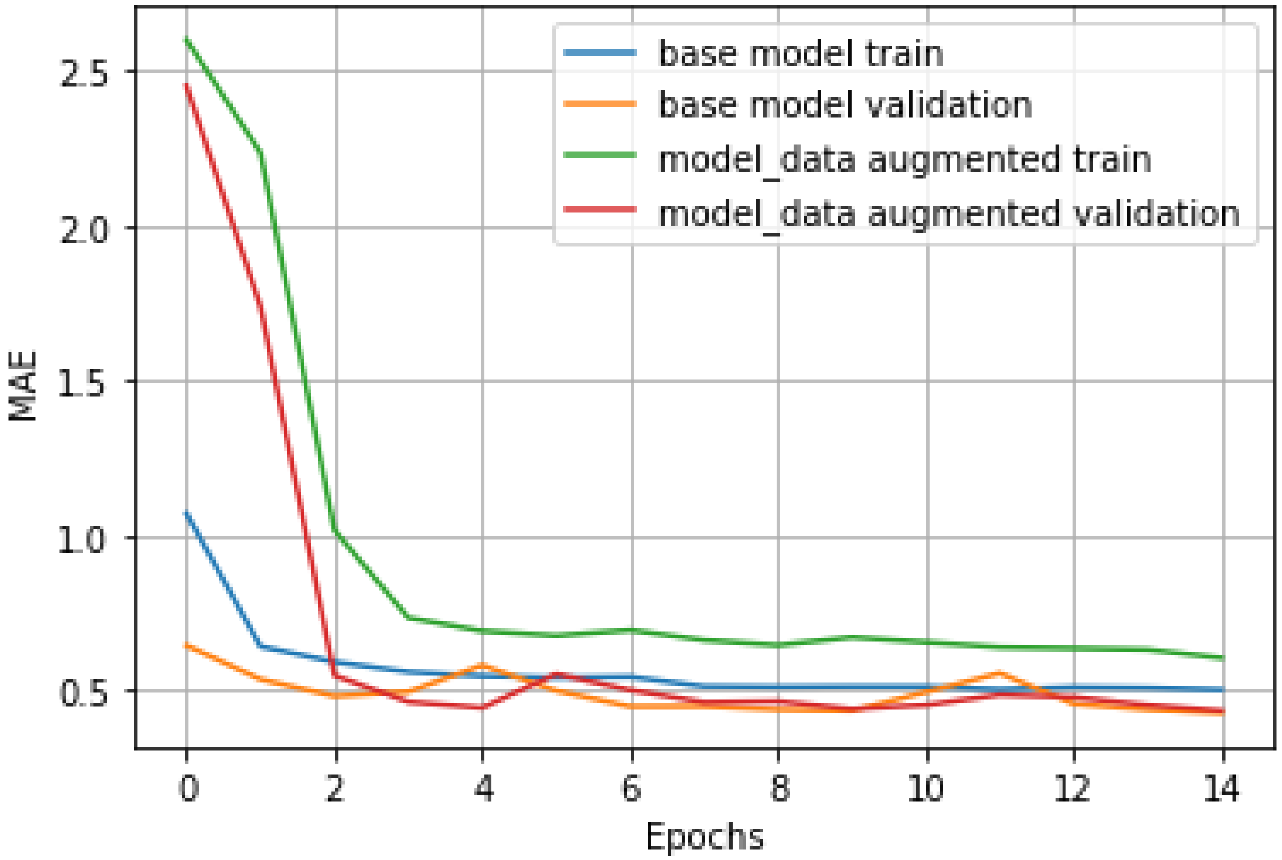


Augmented Dataset Base Model

Dataset has been randomly flipped in vertically,horizontally and converted to grayscale as composition is rather matter of edges than colours.

Augmented Dataset Model Performance

Modest data augementation barely influences the model.



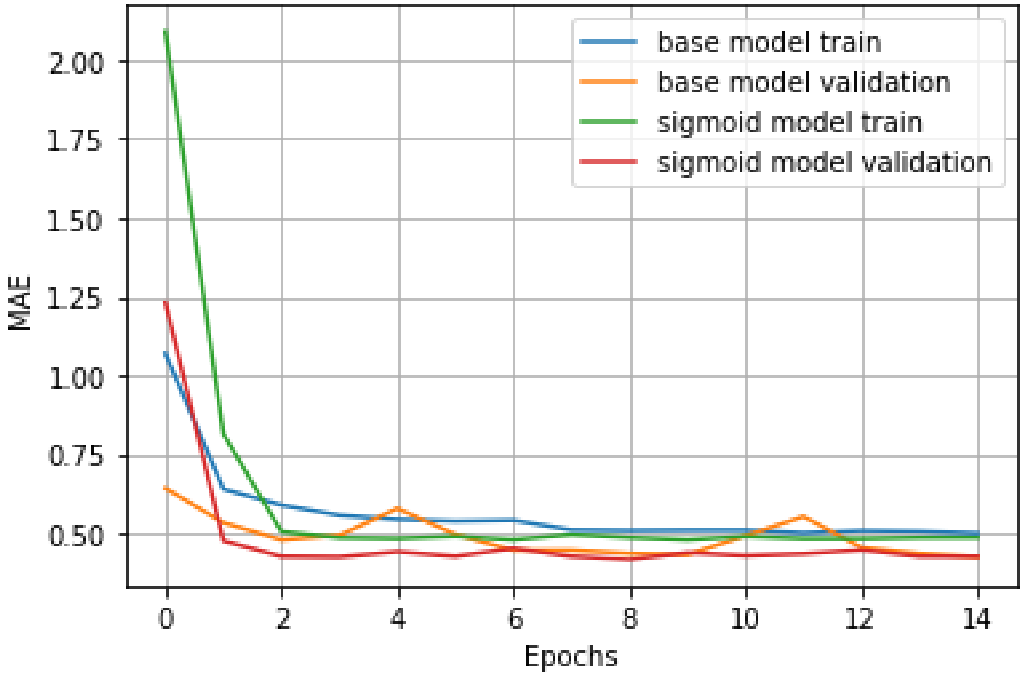
Sigmoidal activations Model

```
model_sig_: tf.keras.models.Model = keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(5, 5), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(7, 7), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dropout(.3),
    tf.keras.layers.Dense(32, activation='sigmoid'),
    tf.keras.layers.Dense(16, activation='sigmoid'),
    tf.keras.layers.Dense(1, activation='linear'),
])

model_sig_.compile(
    optimizer='rmsprop',
    loss = tf.keras.losses.MeanSquaredError(),
    metrics=[tf.keras.losses.MeanSquaredError(), tf.keras.losses.MeanAbsoluteError()])
```

Sigmoidal model performance

Sigmoidal model learns much faster and learning proces is much more stable.



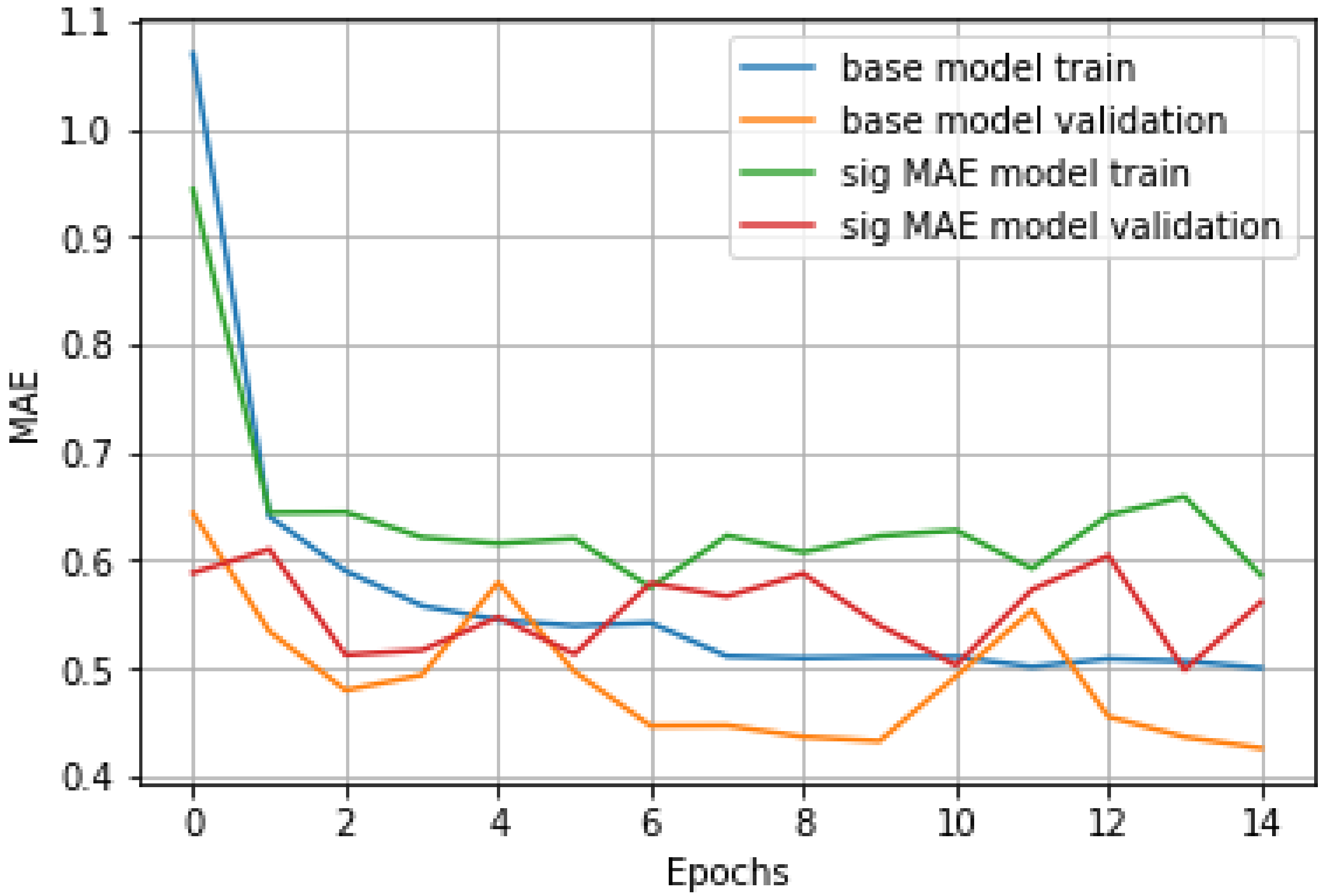
Sig-MAE model

```
model_sig_MAE: tf.keras.models.Model = keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(5, 5), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(7, 7), activation='sigmoid'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dropout(.3),
    tf.keras.layers.Dense(32, activation='sigmoid'),
    tf.keras.layers.Dense(16, activation='sigmoid'),
    tf.keras.layers.Dense(1, activation='linear'),
])

model_sig_MAE.compile(
    optimizer='rmsprop',
    loss = tf.keras.losses.MeanAbsoluteError(),
    metrics=[tf.keras.losses.MeanSquaredError(), tf.keras.losses.MeanAbsoluteError()]
```

SigMAE model performance

Learning with MAE loss function is less stable.



Resume

Using MAE instead od MSE is not prolific.

Dataset may need padding.

0.4 Is the lower bound of what these models may achieve.

Full training

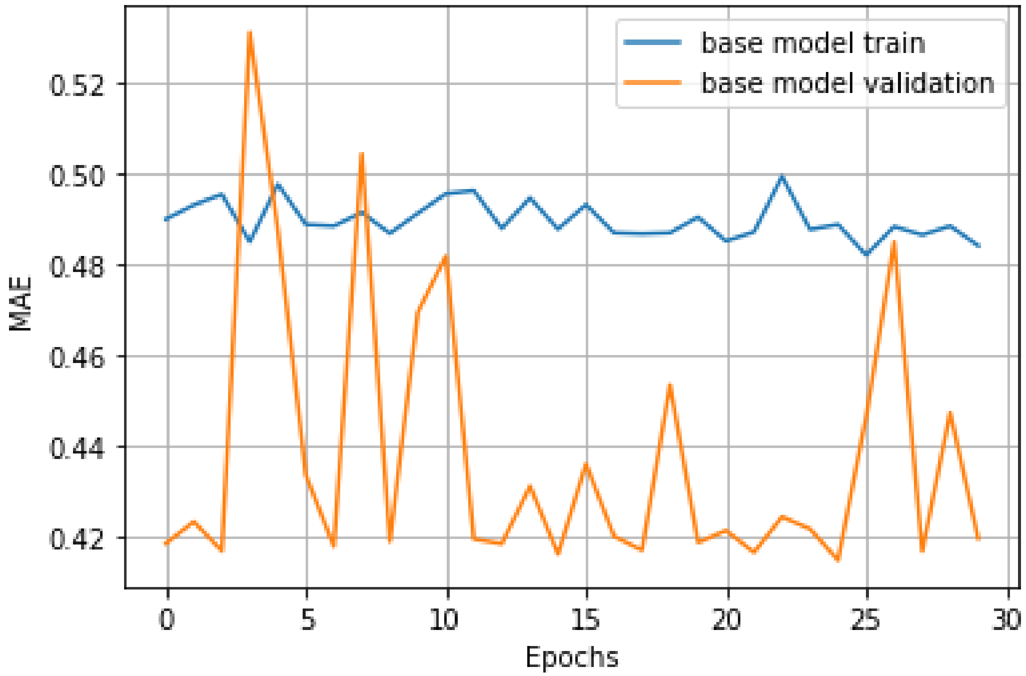
- 8548/10000 Images for training
- 949/10000 Images for validation
- Images resized to (144,144,1) without padding

Chosen architecture

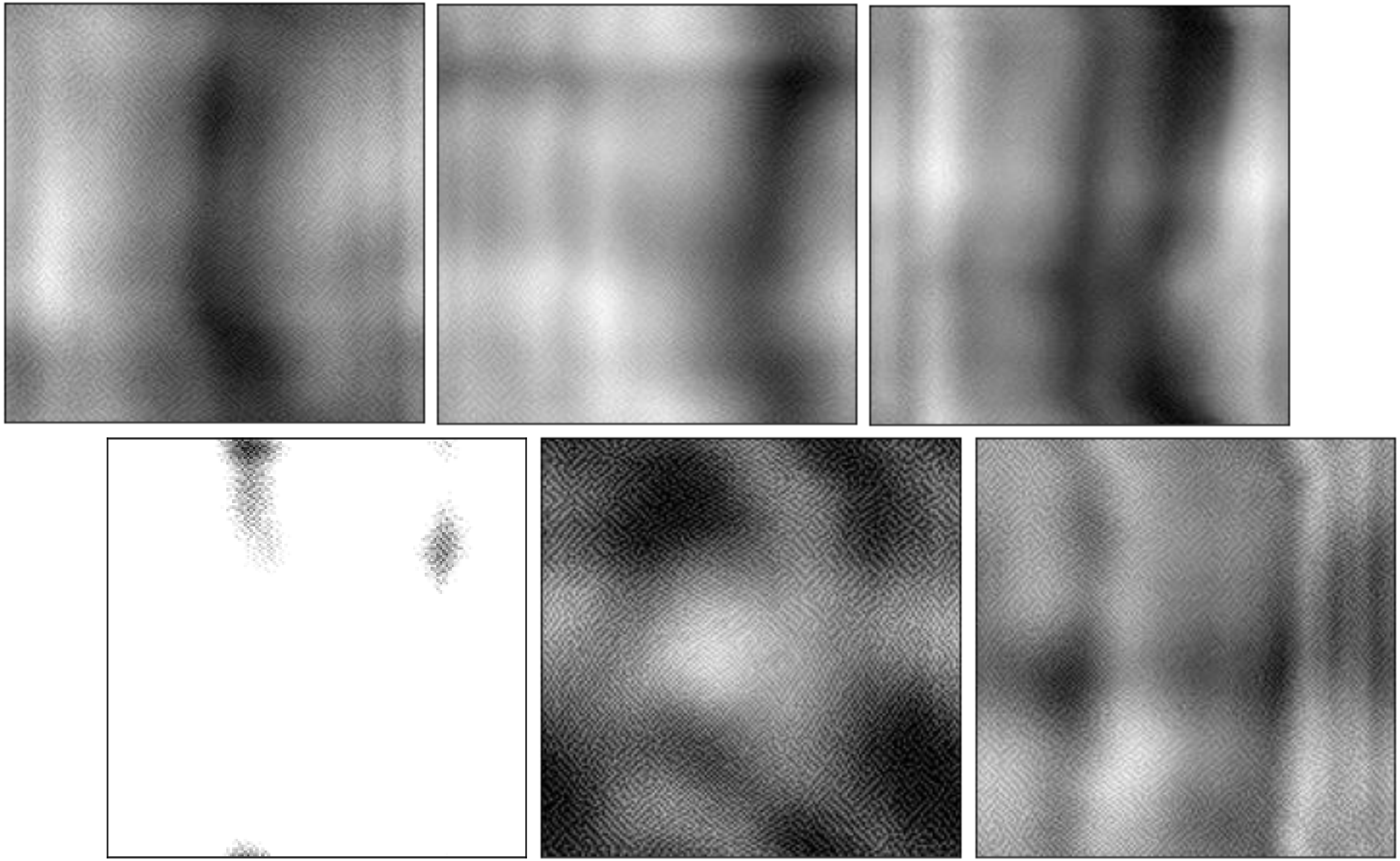
```
model_deeper: tf.keras.models.Model = tf.keras.models.Sequential([
    tf.keras.layers.InputLayer(input_shape=INPUT_SHAPE),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(11, 11), activation='elu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(5, 5), activation='elu'),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='linear'),
])

model_deeper.compile(
    optimizer='adam',
    loss = tf.keras.losses.MeanSquaredError(),
    metrics=[tf.keras.losses.MeanSquaredError(), tf.keras.losses.MeanAbsoluteError()]
)
```

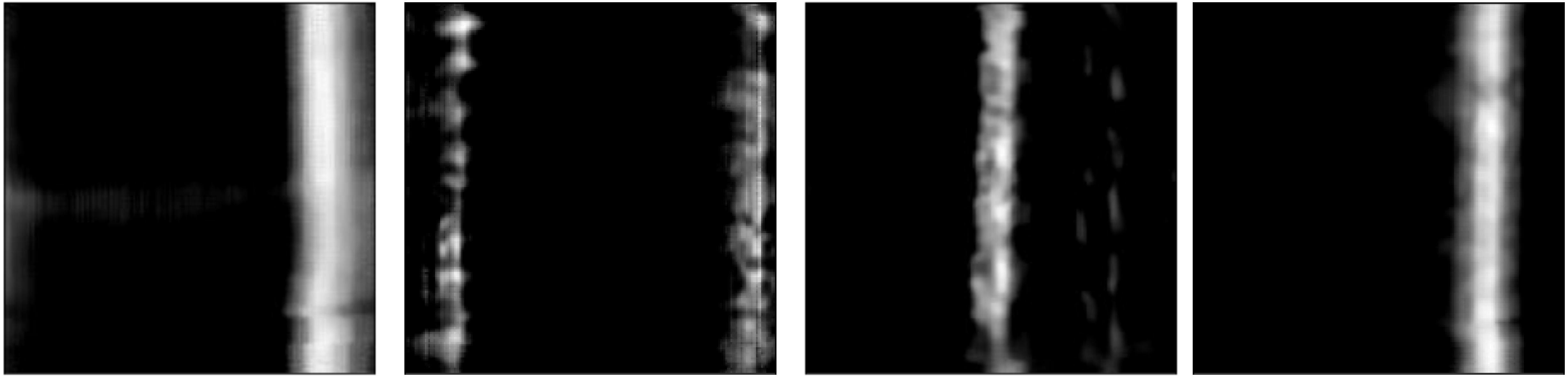
Final training



Chosen activations – first conv layer



Chosen activations max



Final Resume

Error of less than 0.4 MSE is barely achievable, even greater image size is not enough.

Layers max activation visualization shows that horizontal symmetry (left side is reflected to right) is easiest for net to be caught. Second easiest feature of aestheticae is probably rule of thirds and golden ratio.

Future models may achieve lower MAE via separation of flows in convolutional layers.