COMPUTER VISION

Beller Stefan, Binder Lukas, Karasinski Maciej Jacek, Kasper Bianca, Wichser Ilona

Structure

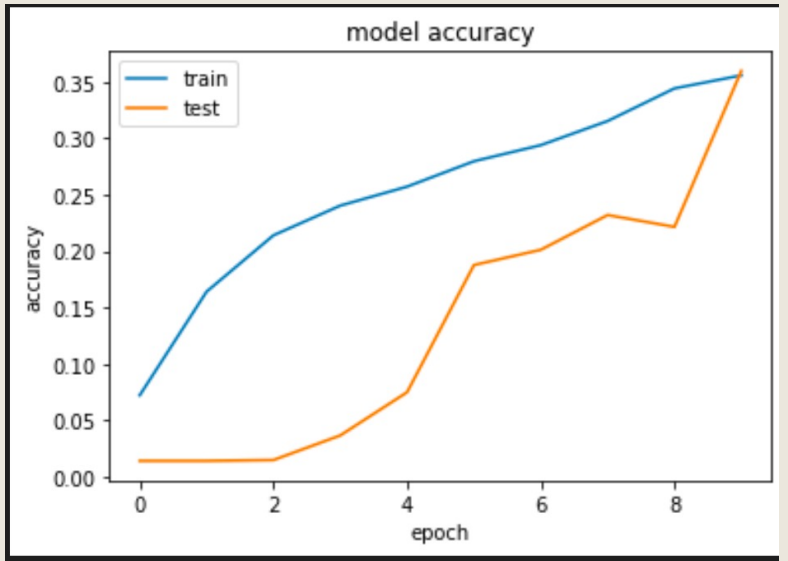
5 TYPES OF MODELS
WE TRAINED

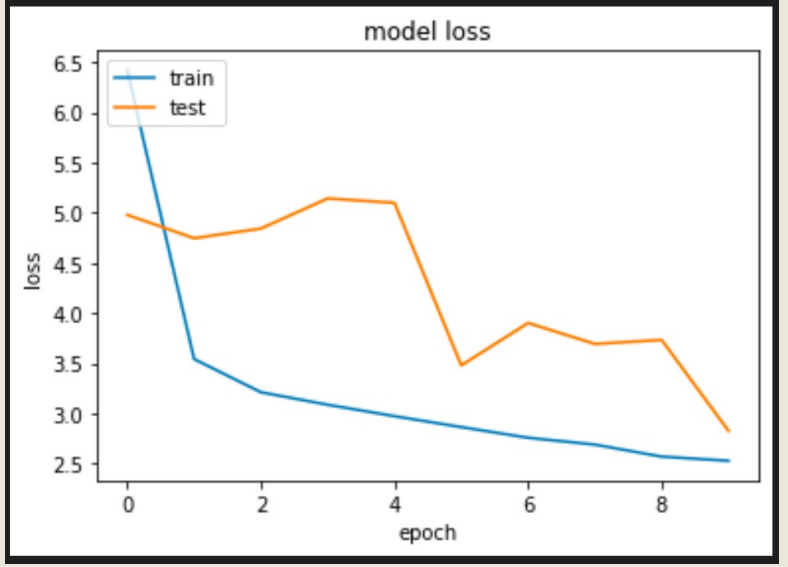
LEARNINGS

Basic Model from Template

- Basic model – first try
- Looks like it would improve in future epochs
- 3x following layers

Conv2D → MaxPool2d → BatchNormalization



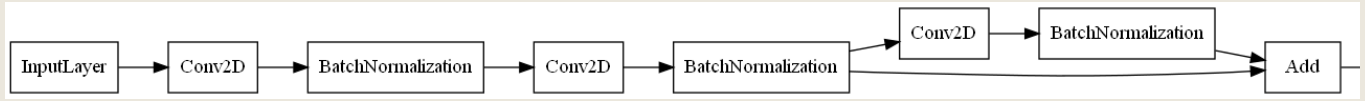


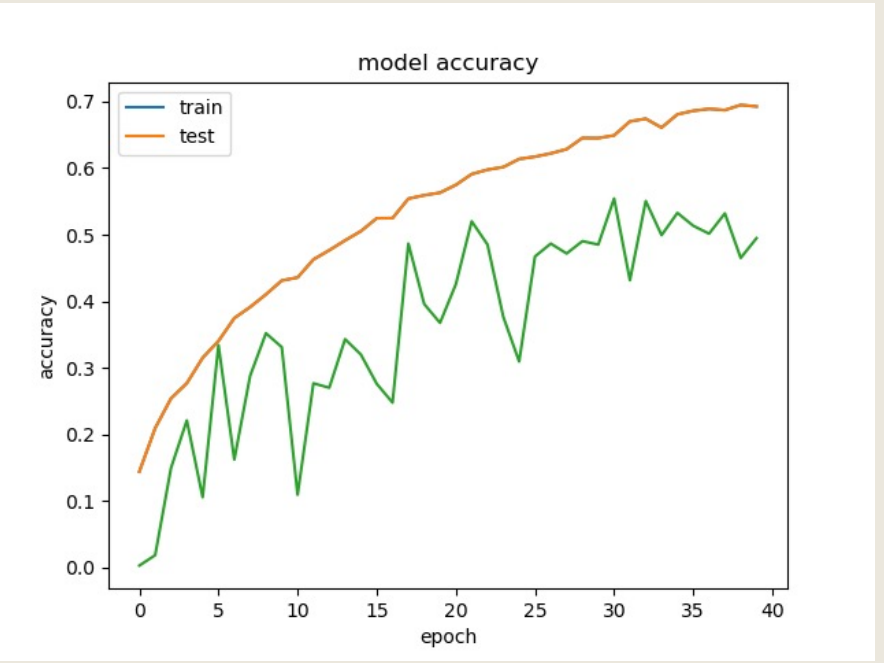
Residual Neural Network

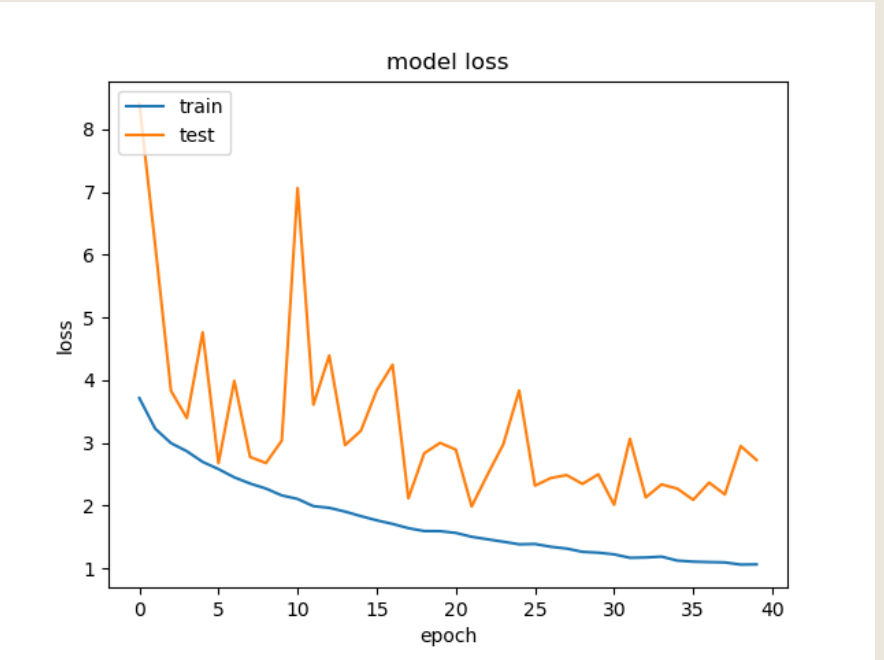
- ResNet was proposed to overcome the problems common CNNs
  - *just stacking convolutional layers to make the model deeper does not guarantee an increase in validation accuracy*
- **ResNet adds the idea of “skip connections”**

First model

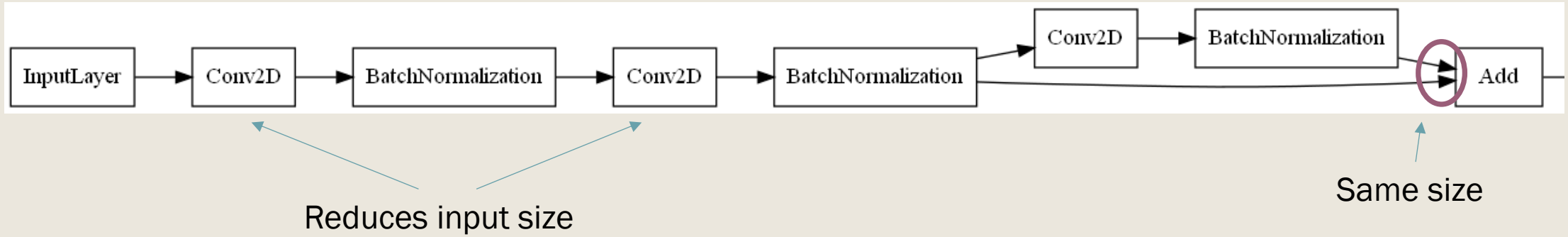
- Simple Residual Neural Network
- ~70s per Epoch on GTX 1080
- ~50% on validation data
- 110k Parameters





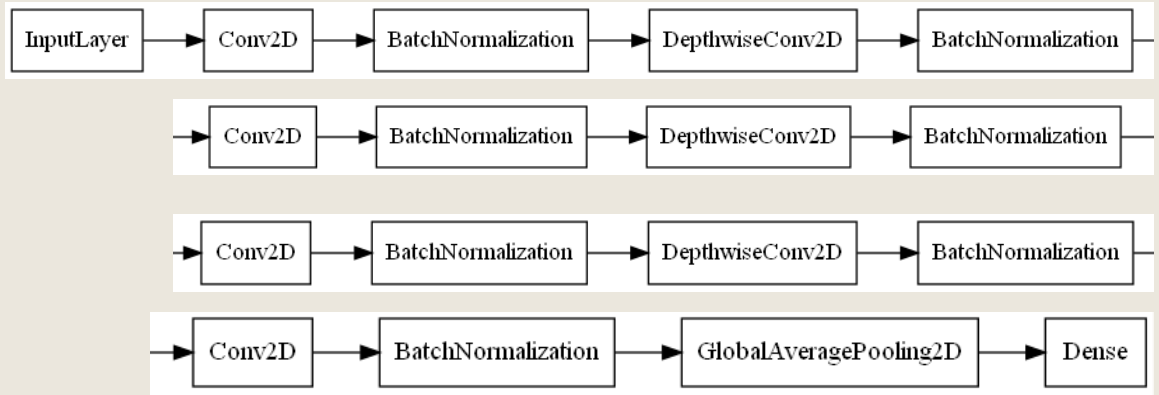


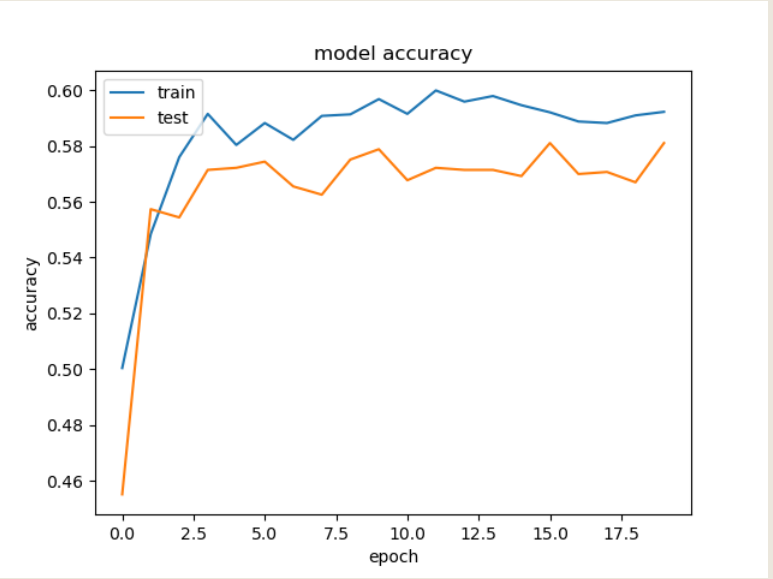
First model - RNN

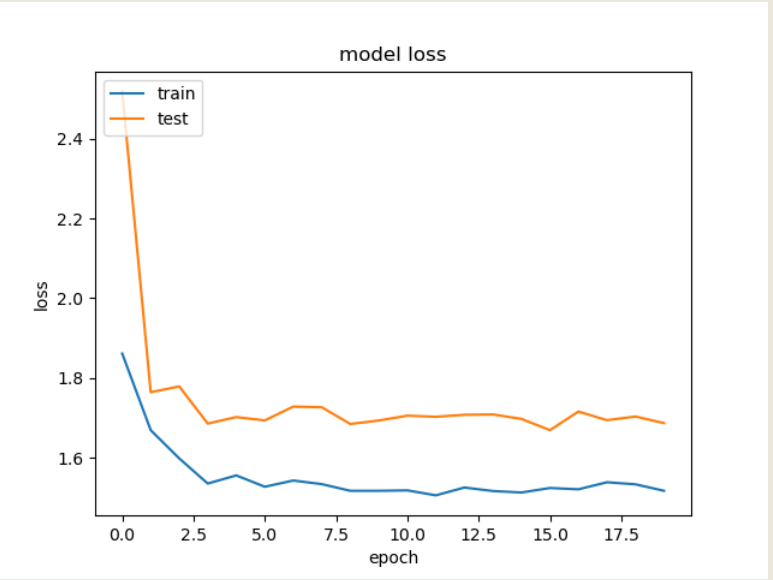


Second Model (MobileNet v1)

- Based on MobileNet v1 → simplified because of number of parameters
- ~70s per Eopoch on GTX 1080
- Addendum: not 1:1 implementation, some different Layers
- ~60% with default filtersize

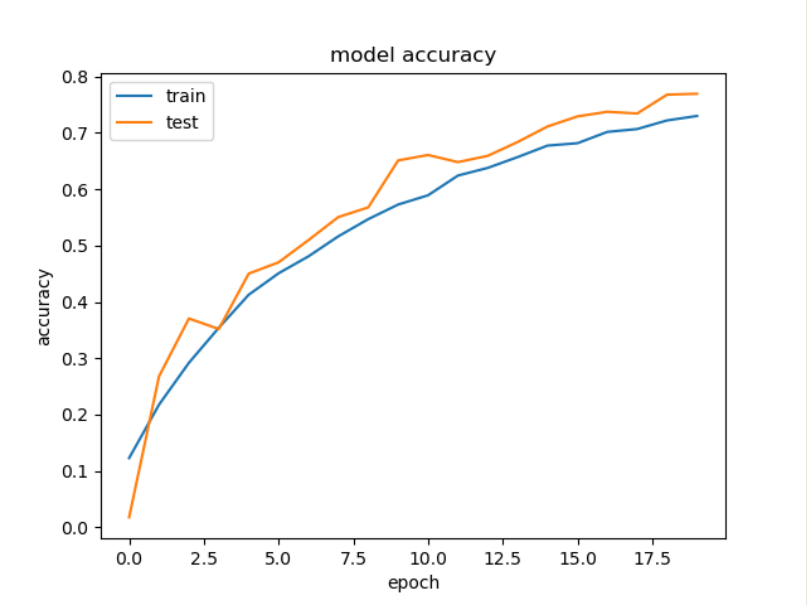


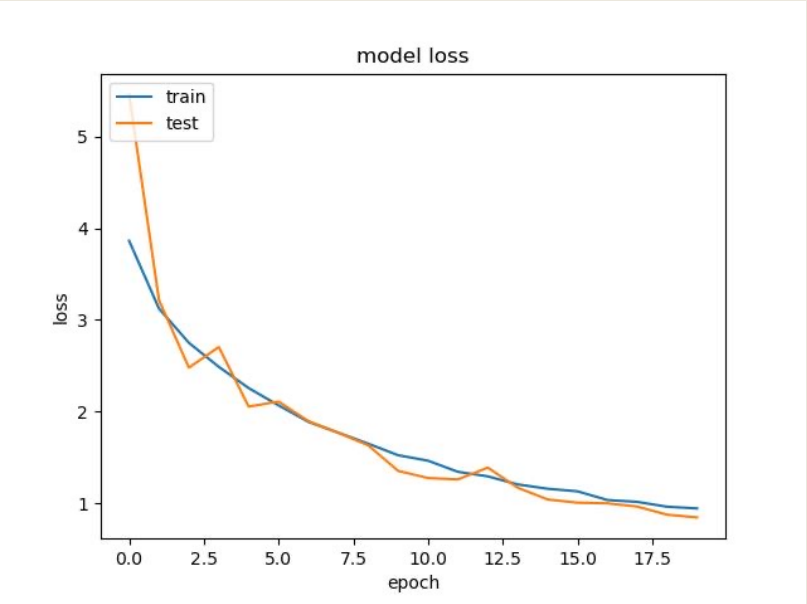


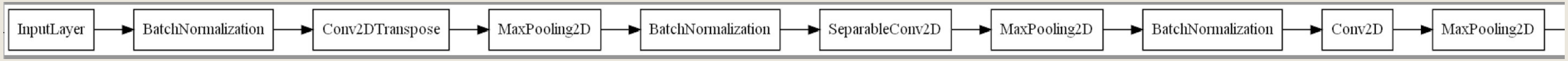


Third Model

- Model inspired by the Model of Nague Marcel
- Added more Layers and bigger Filter sizes → simple Scale up
- ~80% on training data





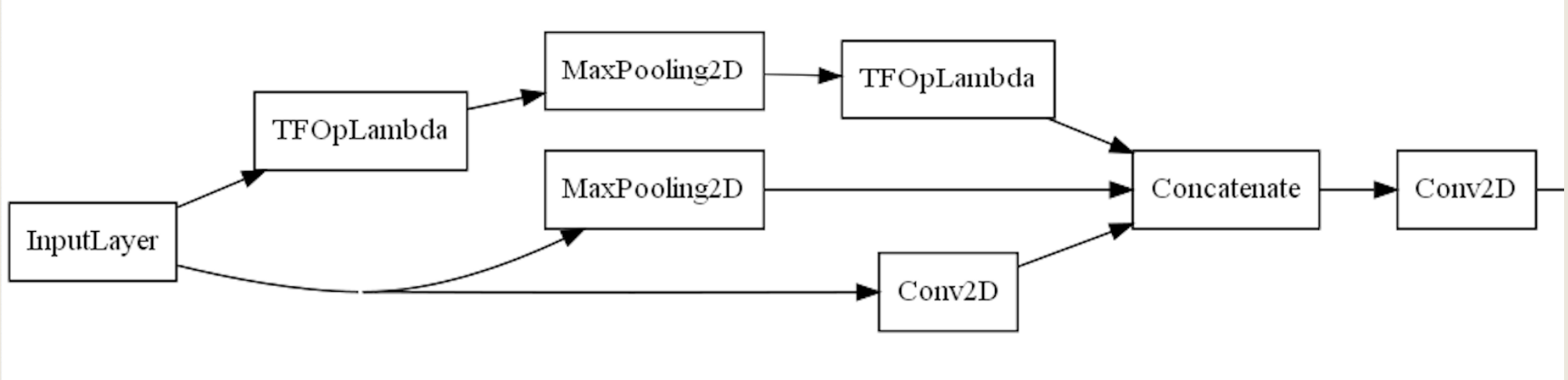


Fourth Model - ExquisiteNetV2

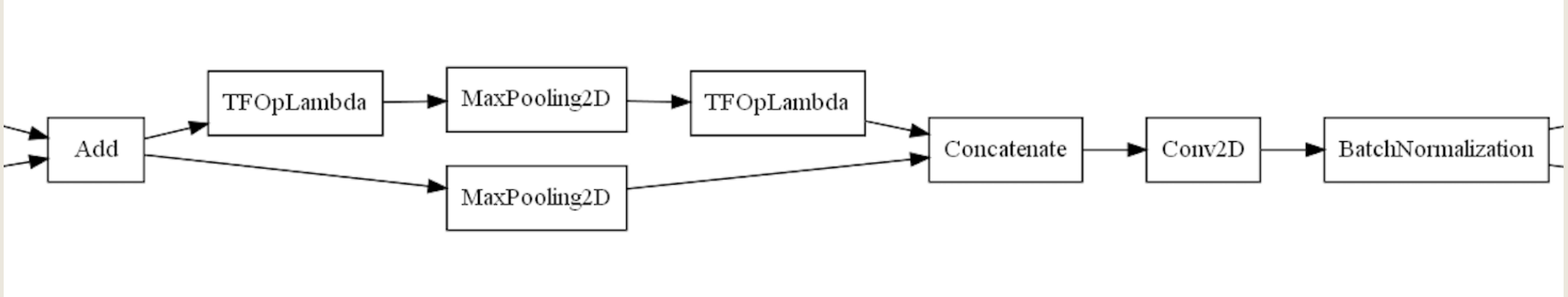
- Customized ExquisiteNetV2
- Tried to convert Pytorch Model to Tensorflow → a lot of Errors
- Didn‘t work at the Start → DepthswiseConv in Pytorch and Tensorflow different
- Accuracy: 93% on test Data
- According to the Paper – Pros - and why we chosen this Net
  - *ExquisiteNetV2 outperforms manny competitors*
  - *ExquisiteNetV2 has fewest amounts of parameters*
  - *Really fast*

https://arxiv.org/ftp/arxiv/papers/2105/2105.09008.pdf

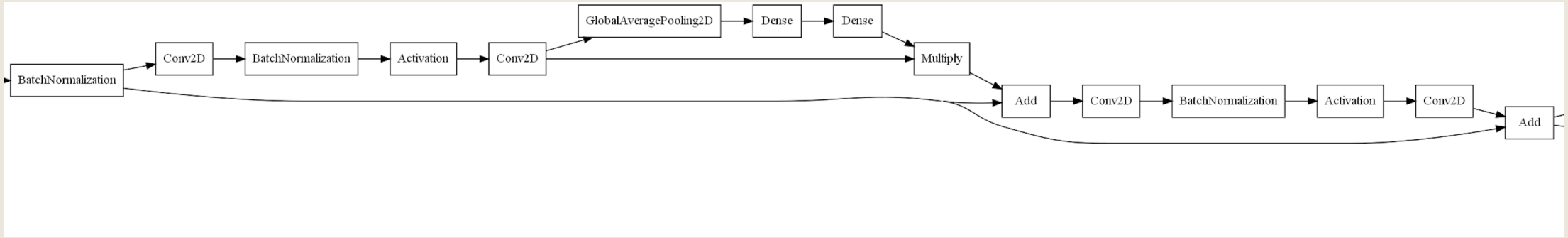
Fourth Model
Feature Concentration



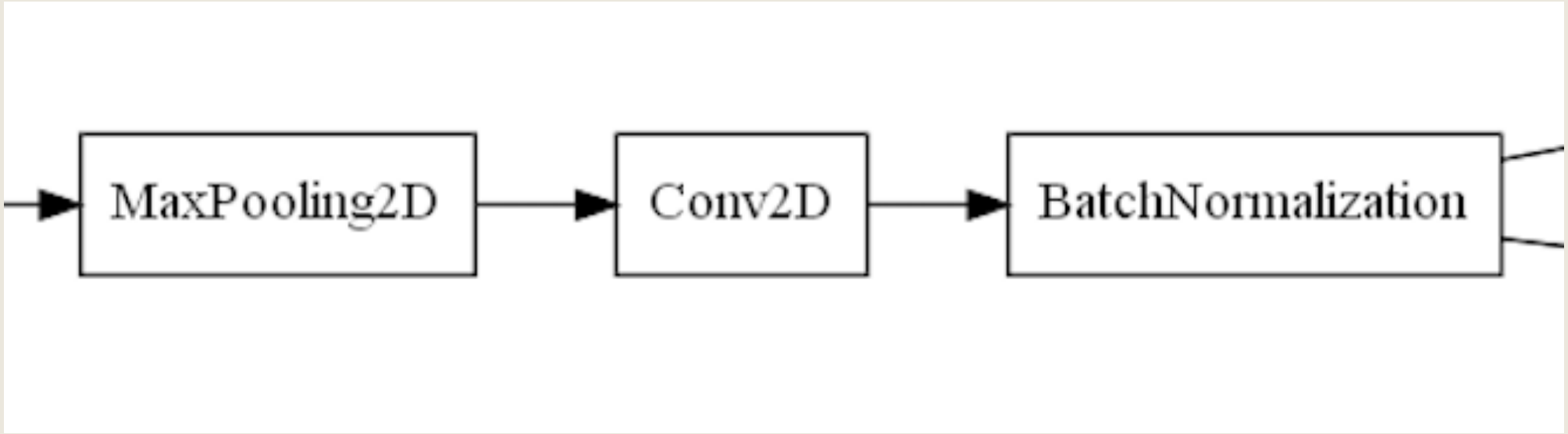
Fourth Model
Concentration Layer



Fourth Model
Feature Extraction Part



Fourth Model
Feature reduction



Fourth Model
Classifier Part

Fourth Model
Result

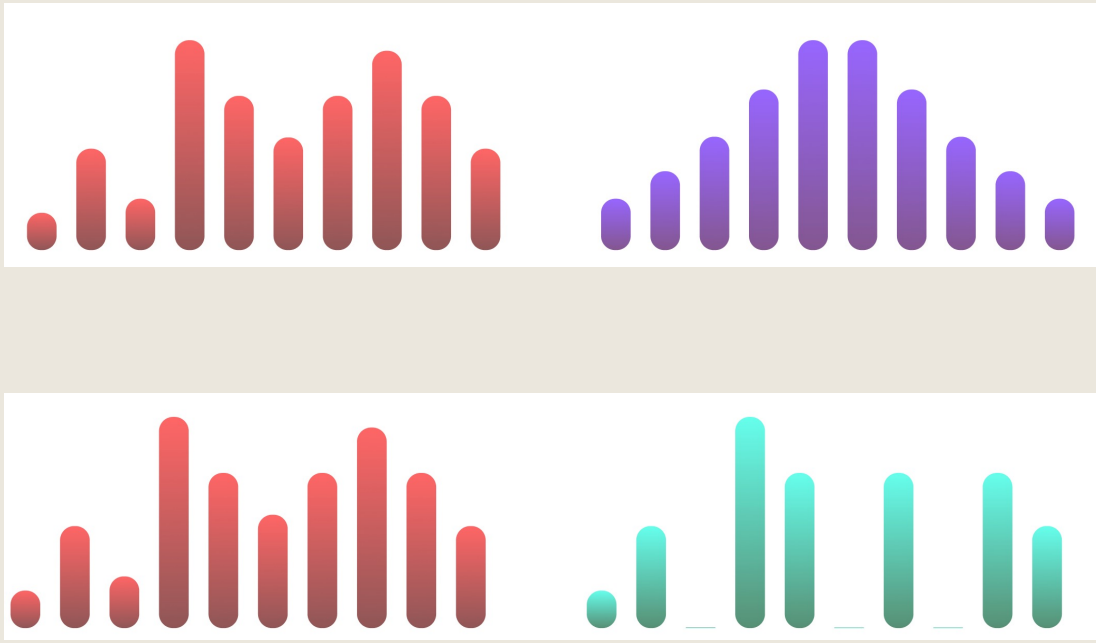
```
results = model.evaluate(datagen_val)
print("test loss, test acc:", results)
```

✓ 28.8s

```
169/169 [==============================] - 29s 169ms/step - loss: 0.4551 - accuracy: 0.9297
test loss, test acc: [0.4550611078739166, 0.9297337532043457]
```

BatchNormalization VS Dropout

- Both are used to prevent overfitting
- BatchNormalization: Normalizes values of the units for each batch
- Dropout: Randomly “drops” a predefined ratio of units
- We had some Networks with a lot of Parameters and low batch size → batch normalization isn’t very good



Learnings

- More is not always better
- Deeper is not always better
- Maybe try smaller learning rate
- Read scientific papers carefully
- Try different approaches
- Don’t stop the training when the first few epochs have bad validation accuracy

Questions ?