Deep Learning

Basic DNN Methods: Batch Normalization

Hochschule Offenburg
University of Applied Sciences

Content

Theory

1. Basics
2. Pros and Cons

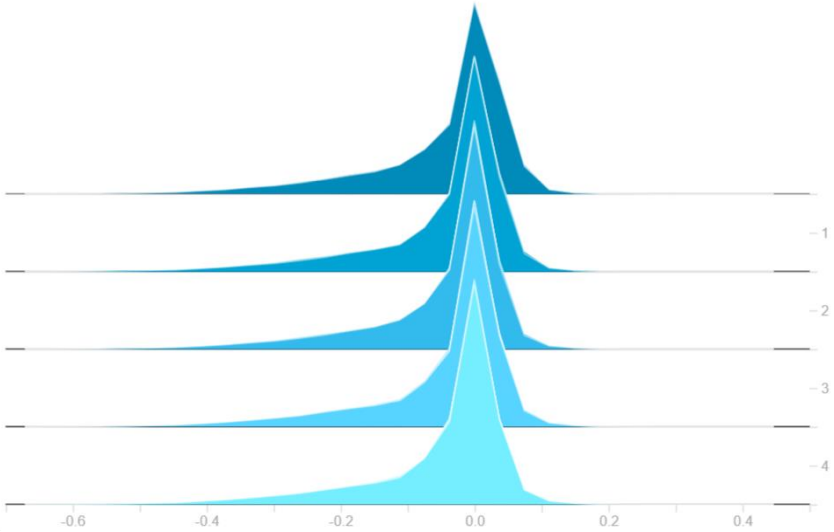
Experiments

1. Setup
2. Training
3. Results

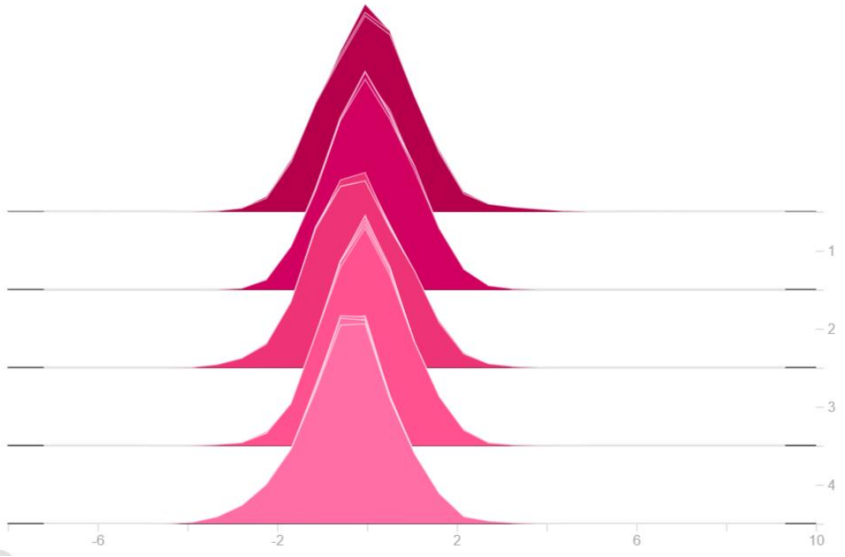
Basics of Batch Normalization

- **Batch Normalization**
  - Normalization of the layer inputs for each training batch
  - Part of the model architecture

Without Batch Norm



With Batch Norm



Basics of Batch Normalization

- To this day, the working mechanism of Batch Normalization is not fully understood
- **Suggested Problem:** „*Internal Covariate Shift*“
  - Distribution of each layer‘s inputs changes during training
  - Slows down the training and makes it hard to train models with saturating nonlinearities
  - Correlation between Internal Covariate Shift and the effects of Batch Normalization not yet proven!

Basics of Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

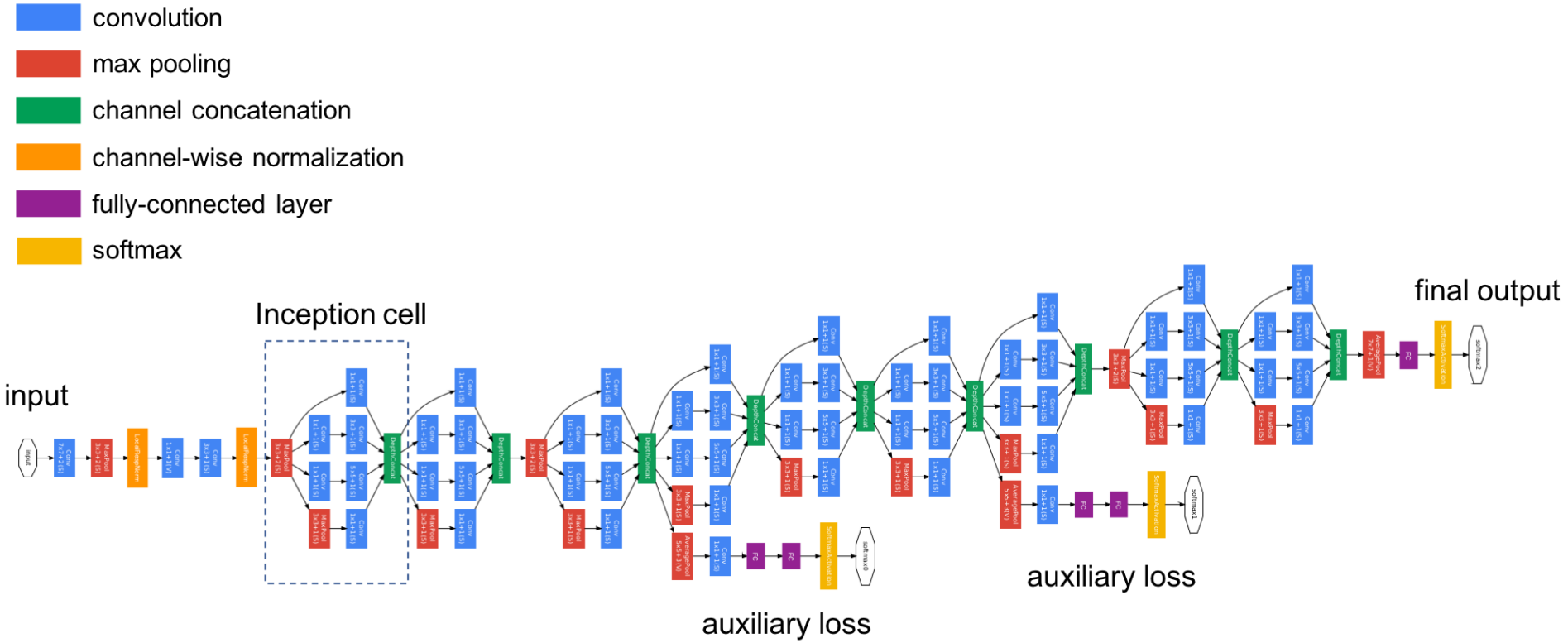
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Pros and Cons of Batch Normalization

- **Pros:**
  - Speeds up training
  - Higher learning rates possible
  - Dropout can be reduced
- **Cons:**
  - More layers and more parameters to learn

Setup: Model Architecture

- Inception Network



https://www.jeremyjordan.me/convnet-architectures/

Setup: Batch Normalization

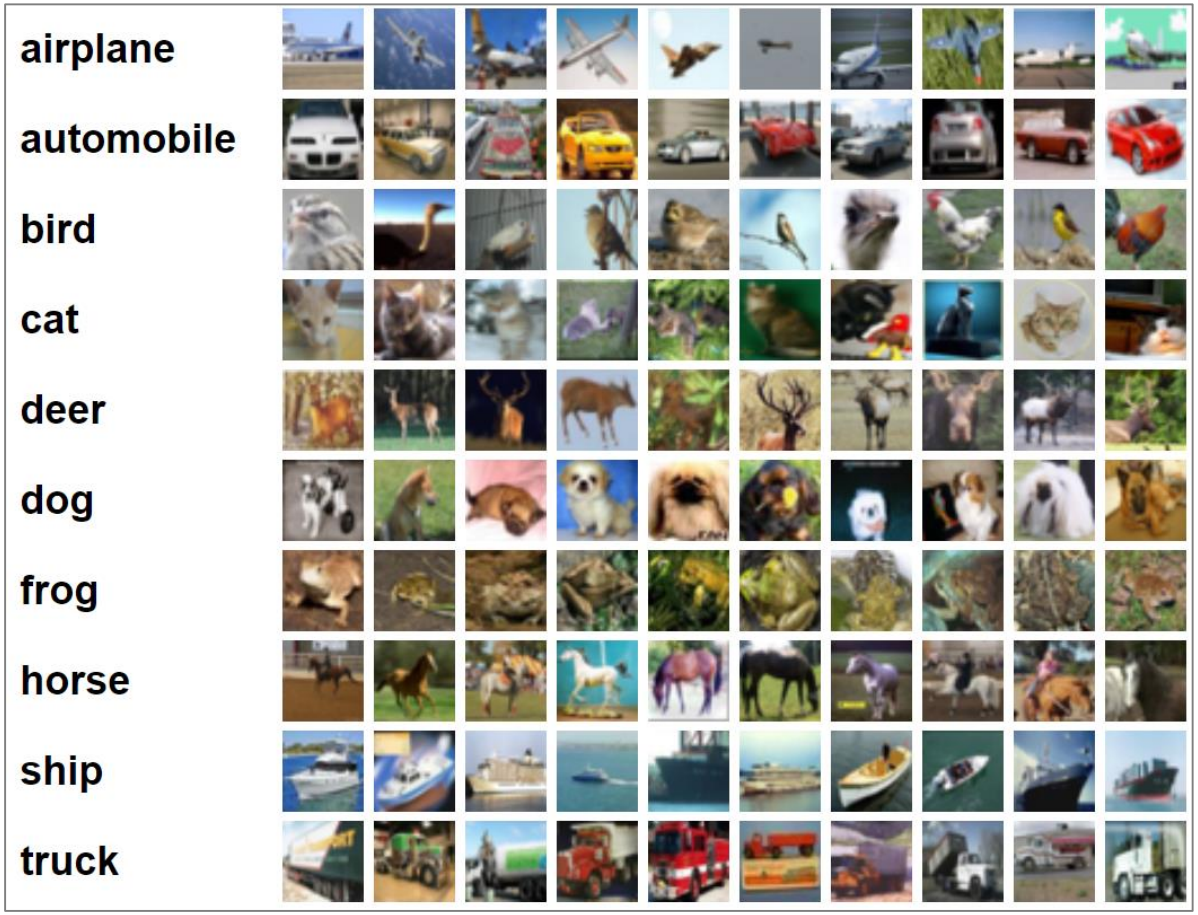
- Batch Normalization is added after Conv and before Activation

```
[ ]  from keras.layers import Conv2D, BatchNormalization, Activation

     # Convolutional Block
     def conv2d(x,filters,kernel_size,strides=1,pad='same',act=True,bn=False):
       # Convolution
       x = Conv2D(filters,kernel_size,strides=strides,padding=pad,use_bias=False)(x)
       # Batch Normalization
       if bn:
         x = BatchNormalization(axis=3,scale=False)(x)
       # Activation
       if act:
         x = Activation('relu')(x)
       return x
```

Training: Data

- **CIFAR-10 dataset**
- 32x32 images
- 10 classes
- 50.000 training
- 10.000 validation

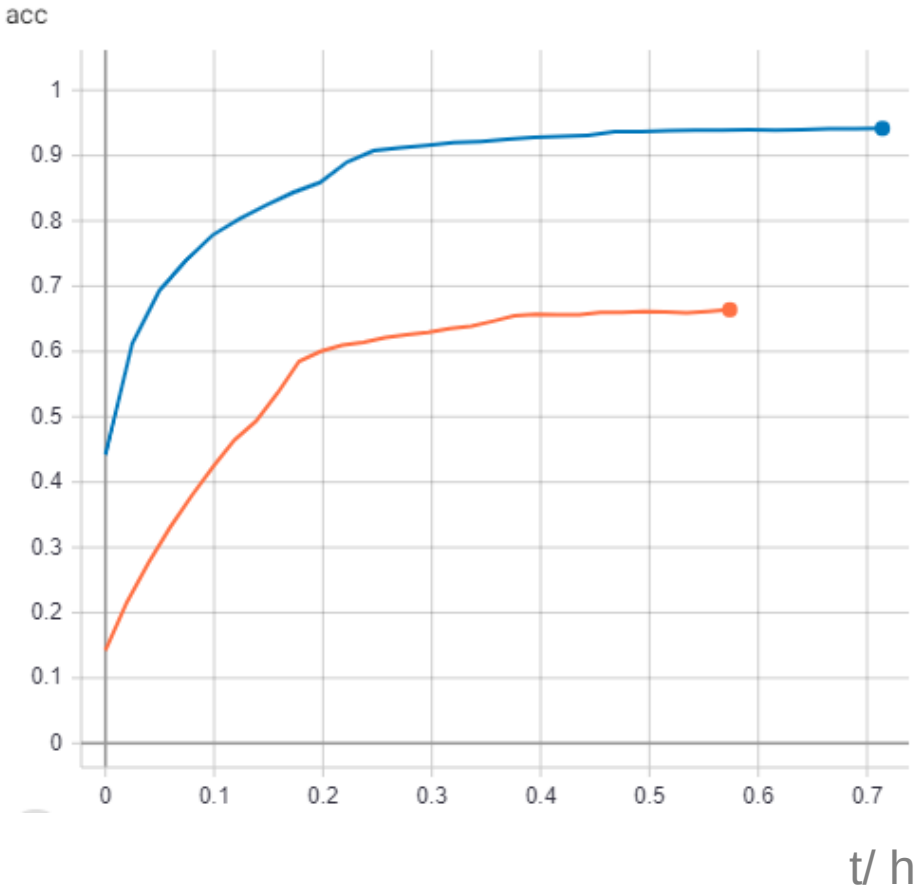


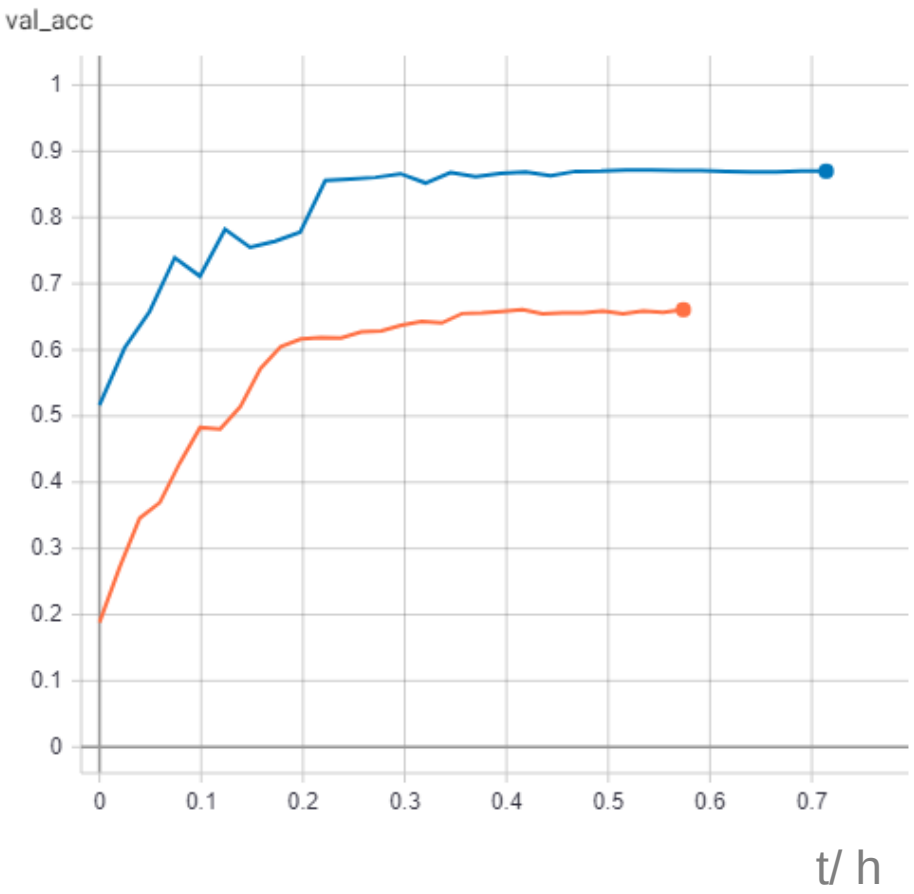
https://www.cs.toronto.edu/~kriz/cifar.html

Training: Parameters

Number of epochs	30
Batch size	250
Initial learning rate	0.01
Decay	10% drop every 10 epochs
Optimizer	SGD
Dropout	none
Data augmentation	horizontal flip, shearing, zooming

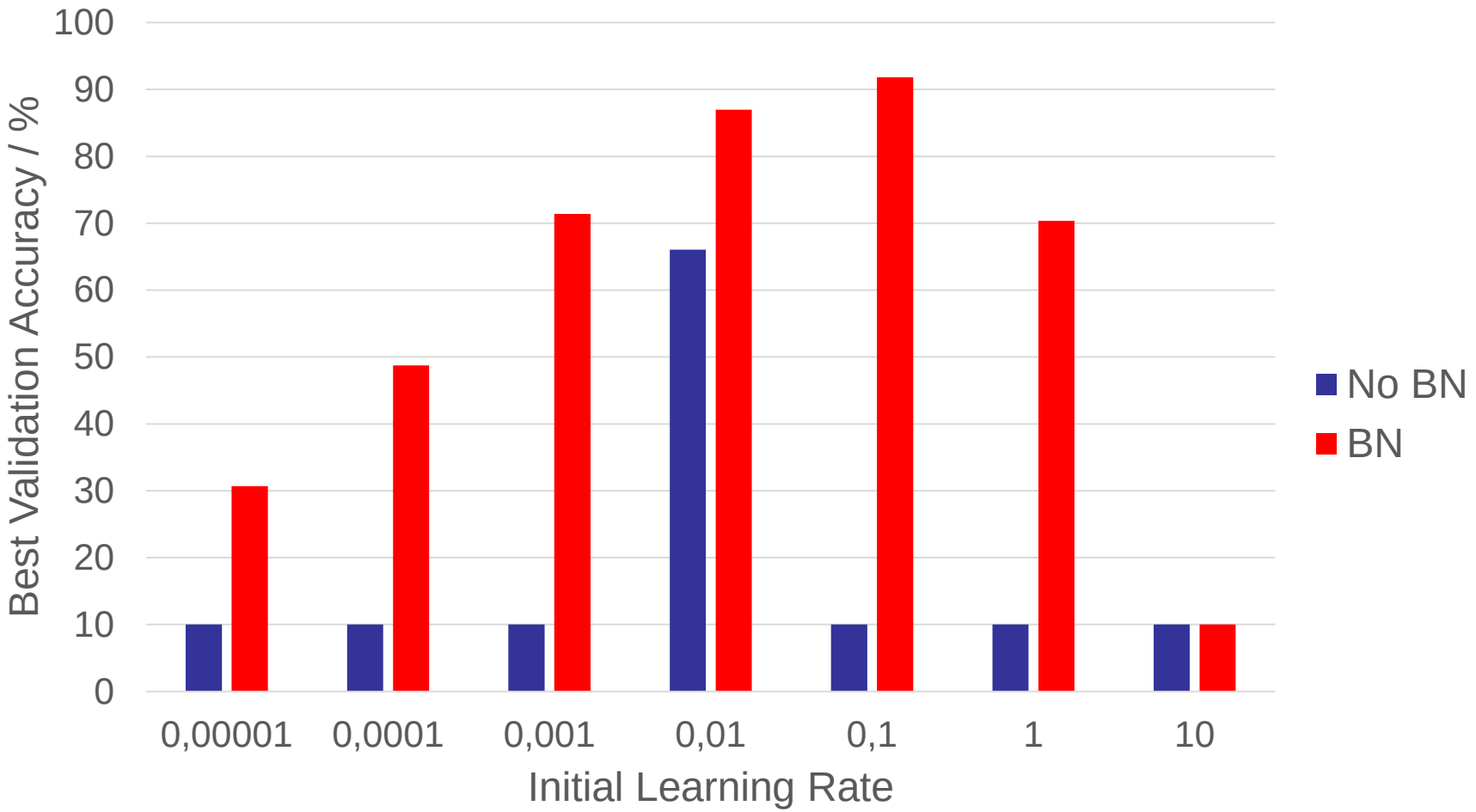
Results



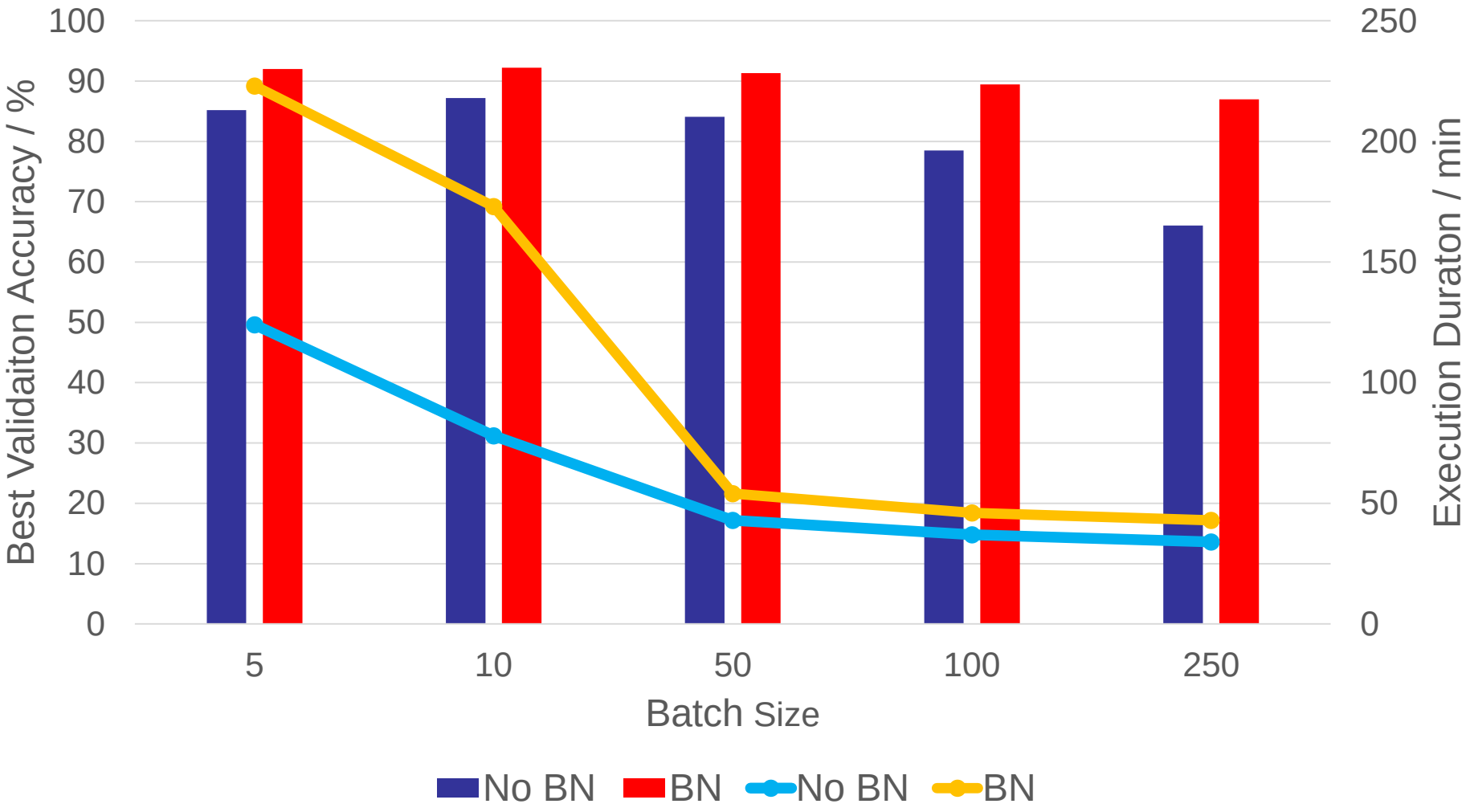


- **BN**
- **No BN**

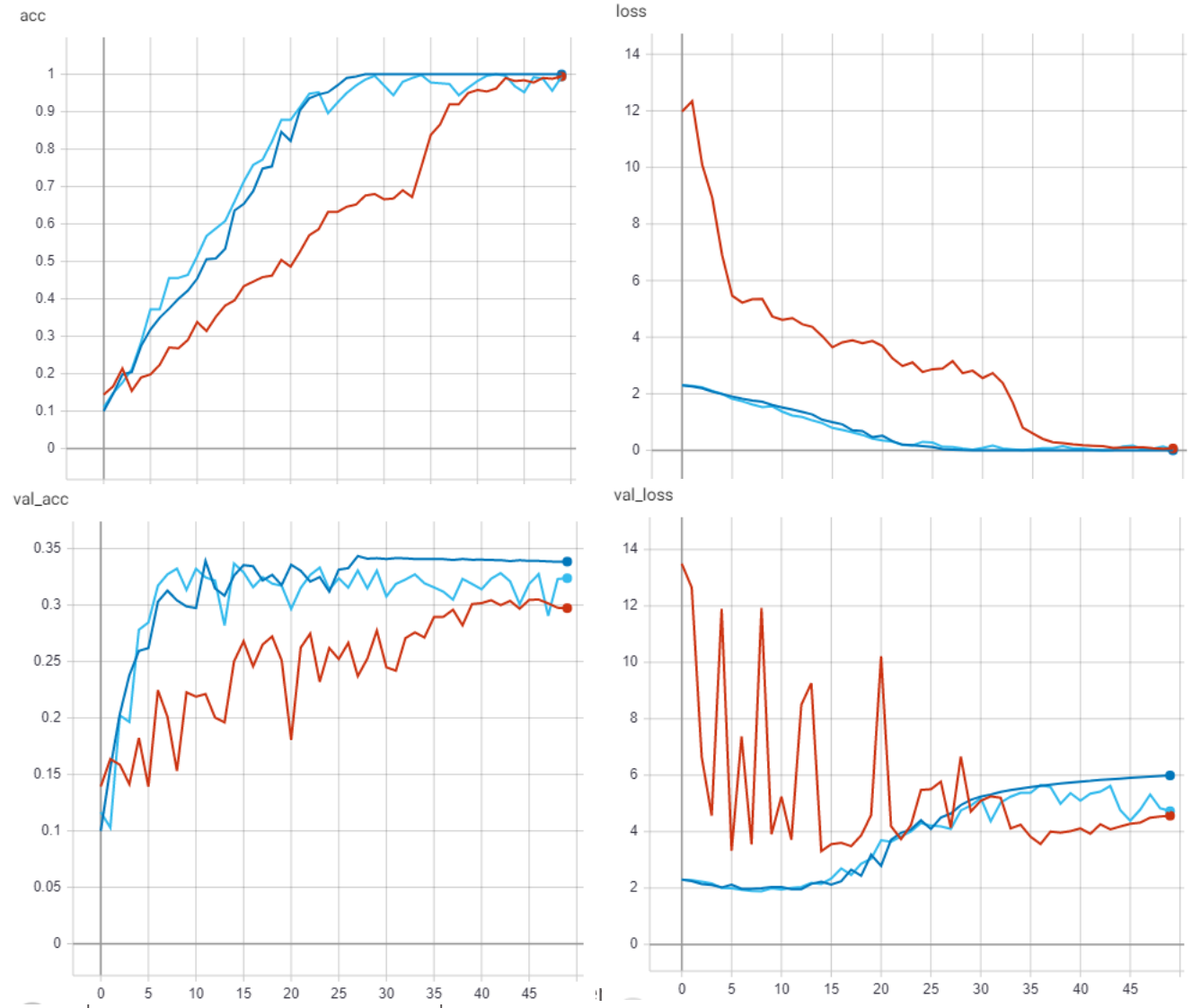
Results: Learning rates



Results: Batch Sizes



Results: Dropout Comparison



- No BN
- No BN with 50% Dropout
- BN

Note:

These experiments were conducted with a simpler model and fewer training images to highlight the effects of dropout

Summary

- The experiments show that Batch Normalization can
  1) increase the validation accuracy and training speed of a model
  2) extend the execution time per step
  3) allow the use of both higher and smaller learning rates
  4) work good with medium batch sizes (50-100)
  5) regularize the model, similar to dropout

Bibliography

- Sergey Ioffe and Christian Szegedy. „Batch normalization: Accelerating deep network training by reducing internal covariate shift“, 2015.

  https://arxiv.org/abs/1502.03167

- Code used for experiments

  https://github.com/KieferN/Batch_Normalization