Effect of Batch Normalization and Dropout in Deep Learning Networks

ADEIKALAM Pierre
CHEN Guangyue
XU Kevin

Introduction

Aim

Batch Normalization and Dropout are standard techniques for the regularisation of deep networks.

The aim of this presentation is to explain intuitively how these two techniques work, what problem they aim to solve and how to use them effectively. To this end, this presentation has been structured as follows :

1. Mechanism of Batch Normalization and its Effects
2. Mechanism of Dropout and its Effects
3. Illustrative Experiments

Normalization

Normalization is the traditional technique of setting the empirical mean of a dataset to 0 and its variance to 1.

If we let :

- $x_1, ..., x_n$ be a dataset of n observations
- μ be its mean
- σ^2 be its variance

Then normalizing this dataset corresponds to the operation :

$$\forall i \in \{1, ..., n\} \qquad \hat{x}_i \leftarrow \frac{x_i - \mu}{\sigma}$$

Batch Normalization

In Deep Learning, the aim of normalizing the outputs of each layer of neurons is to :

- Hopefully speed up training.
- Make sure each layer is training on the same distribution of inputs to avoid sharp local minimas.
- Make sure no neuron is "dead", i.e never activates.

Since we train our network by batches, the outputs of a layer for a specific sample can only be normalized with respect to the other samples in the batch.

Batch Normalization

Algorithm 1 Batch Normalization Forward Operation

Input : Batch of inputs $(x_1, ..., x_n)$
Output : Batch of normalized inputs $(\hat{x}_1, ..., \hat{x}_n)$

procedure BatchNormalization$(x_1, ..., x_n)$

$\hat{\mu} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$ ▷ Mini-batch empirical mean

$\hat{\sigma}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \hat{\mu})^2$ ▷ Mini-batch empirical variance

$\forall i, \hat{x}_i \leftarrow \gamma \frac{x_i - \hat{\mu}}{\hat{\sigma}} + \beta$ ▷ Normalization of the inputs

γ and β are trainable parameters that give the Batch Normalization operation the flexibility to become the identity function if normalization is suboptimal.

Speed up in Training

To understand how Batch Normalization trains a network faster, we look at the simple case of a "dead" neuron which often happens when using the ReLU activation function.

$$\mathrm{ReLU}(x) = x1_{x>0}$$

The gradient of this function is constant and it avoids the problem of vanishing gradient.

Speed up in Training

The problems of this activation function are :

- The gradient of the ReLU function is 0 for $x < 0$.
- If poorly initialized, a neuron can almost always output 0.
- If a neuron almost always outputs 0, it very rarely updates.
- More passes are needed to train this neuron.

Speed up in Training

However, if we normalize the outputs of this neuron :

- The neuron will output a positive value for about half of the samples.
- Necessarily, the neuron will update every iteration of Mini-Batch SGD.
- The neuron will never die.

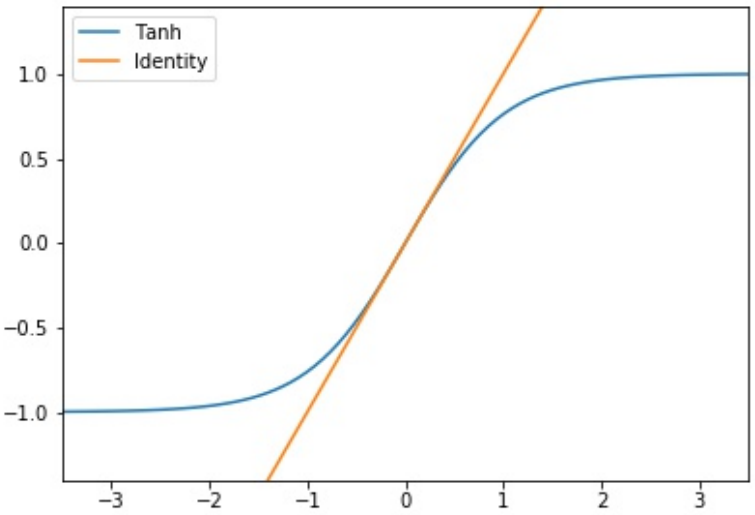
Negative effects

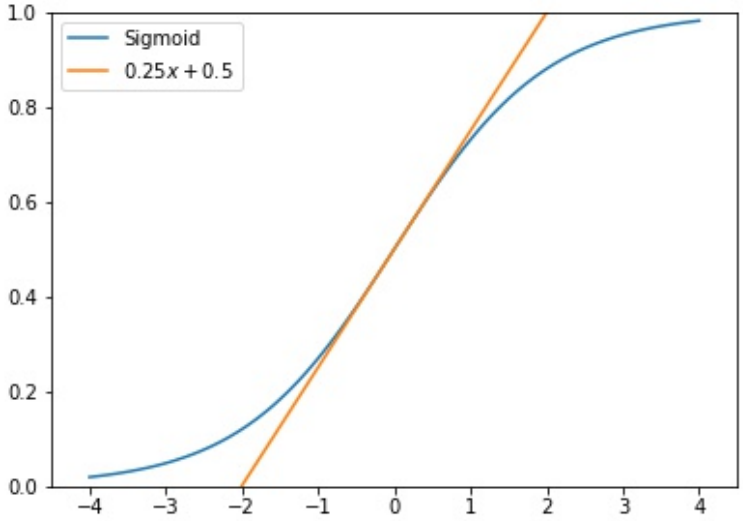
The problem with Batch Normalization appear when we use the activation functions tanh and sigmoid.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$S(X) = \frac{1}{1 + e^{-x}}$$

Negative effects





These functions are similar to the identity function near 0. By rescaling the outputs with Batch Normalization, we could end up canceling the non-linearity of the functions.

Regularization

- At each epoch, the normalization is computed with different parameters since the samples in the mini-batch change.
- This adds noise to the hidden states of the network.
- Intuitively this acts as a data augmentation technique, thus regularizing the network.

Motivation of Dropout

The objective of Dropout is to simulate ensembling methods.

- Ensemble methods combine a multitude of weak regressors into a strong, more powerful regressor.
- This approach helps combat overfitting for models with a very large capacity, such as neural networks or deep decision trees.
- Random Forest constructs many Decision Trees with a random selection of the features, and averages their individual predictions.

Mechanism

If we imagine that a neuron is a regressor,

- Dropout consists in setting a random amount of the inputs of this neuron to 0 and perform a training step.
- This effectively trains this neuron with a random subset of its "features".
- At test time, the weights of this neuron are re-scaled to "average" the outputs of the previous layer.

Mechanism

Another way to see this mechanism :

- At each iteration, we sample a thinner version of the network and train it for a single step.
- If a network has n neurons, the final prediction is the averaged prediction of up to 2^n thinned versions of this network.

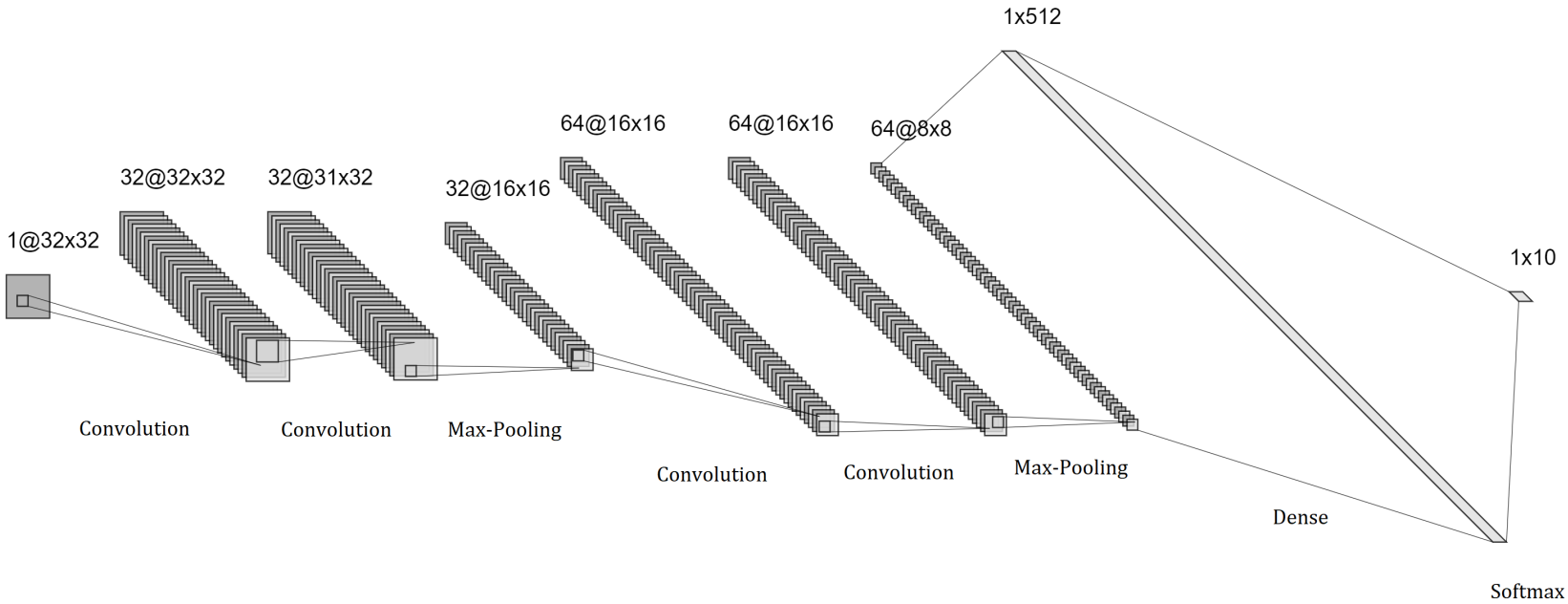
Preventing co-adaptation

Co-adaptation

- When a network is trained, the magnitude of the weights associated with a neuron depend on the frequency of its activations.
- When active, neurons fight for relevancy by shifting the magnitude of their weights up.
- A balance is reached when neurons have "co-adapted" to each other by setting their weights relatively low or high depending on the frequency of their activation.

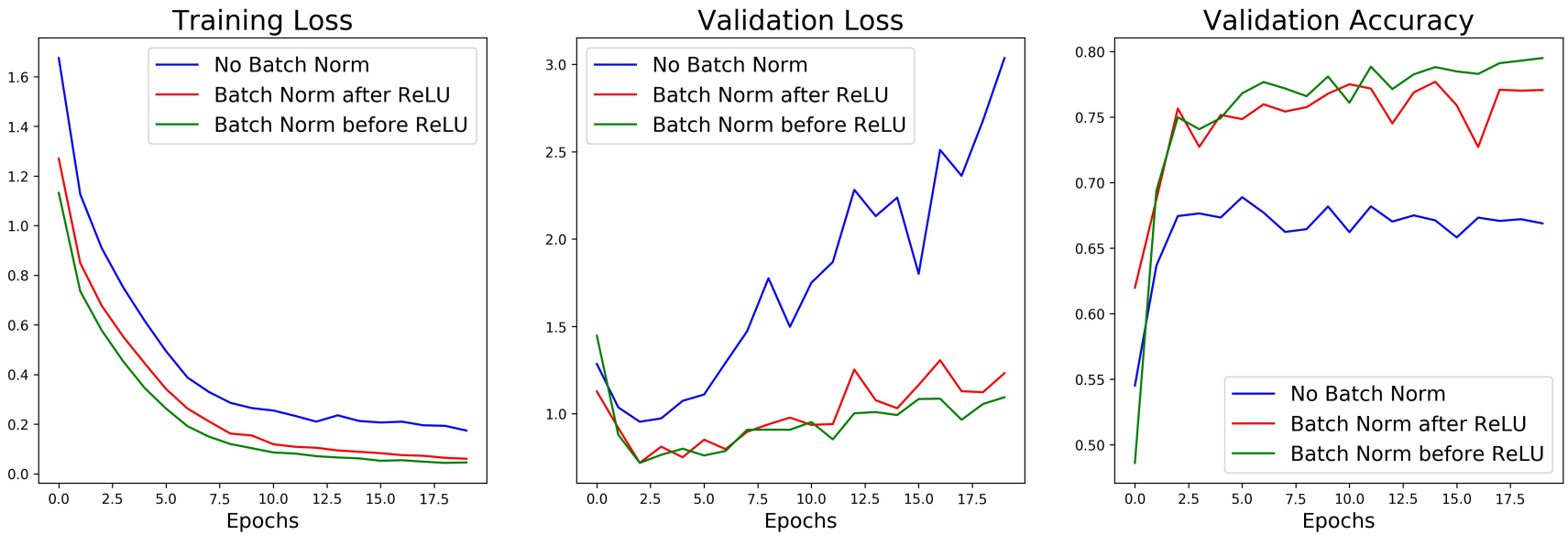
With Dropout, each neuron has to learn a relevant feature such that taking a sample of them leads to an overall good prediction. Each neuron has to learn independently of the others.

Experiment 1 : A simple CNN Model



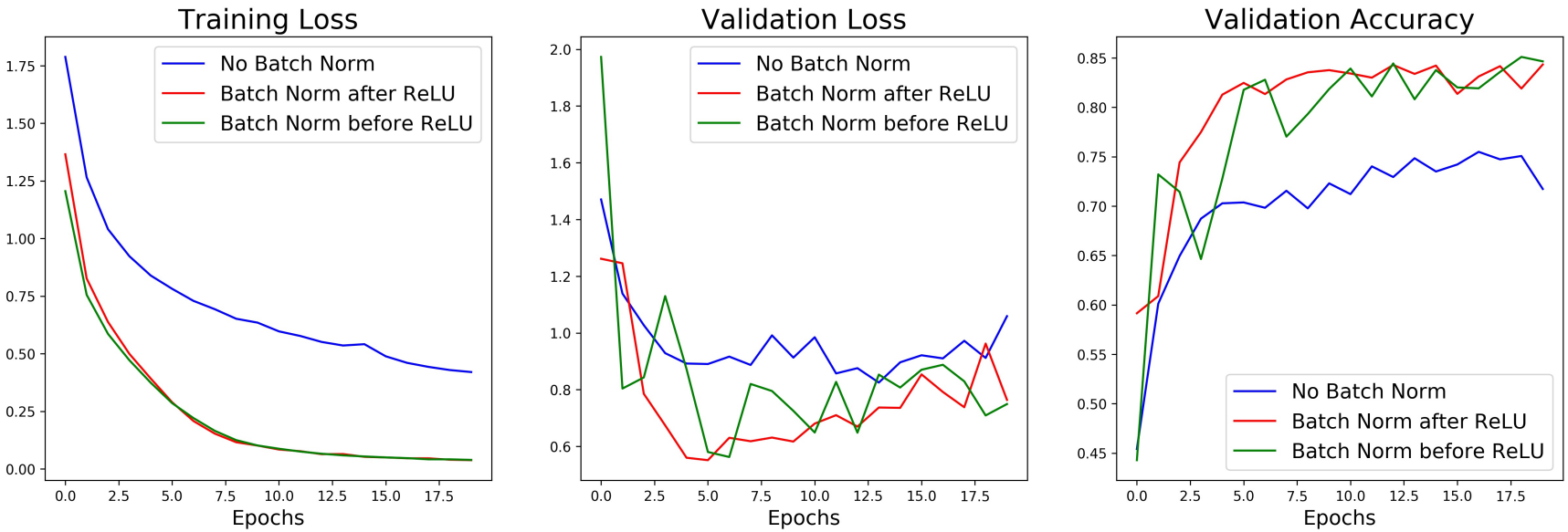
For these experiments, we train different architectures on the CIFAR-10 image classification dataset.

BN Experiment 1 : A simple CNN Model



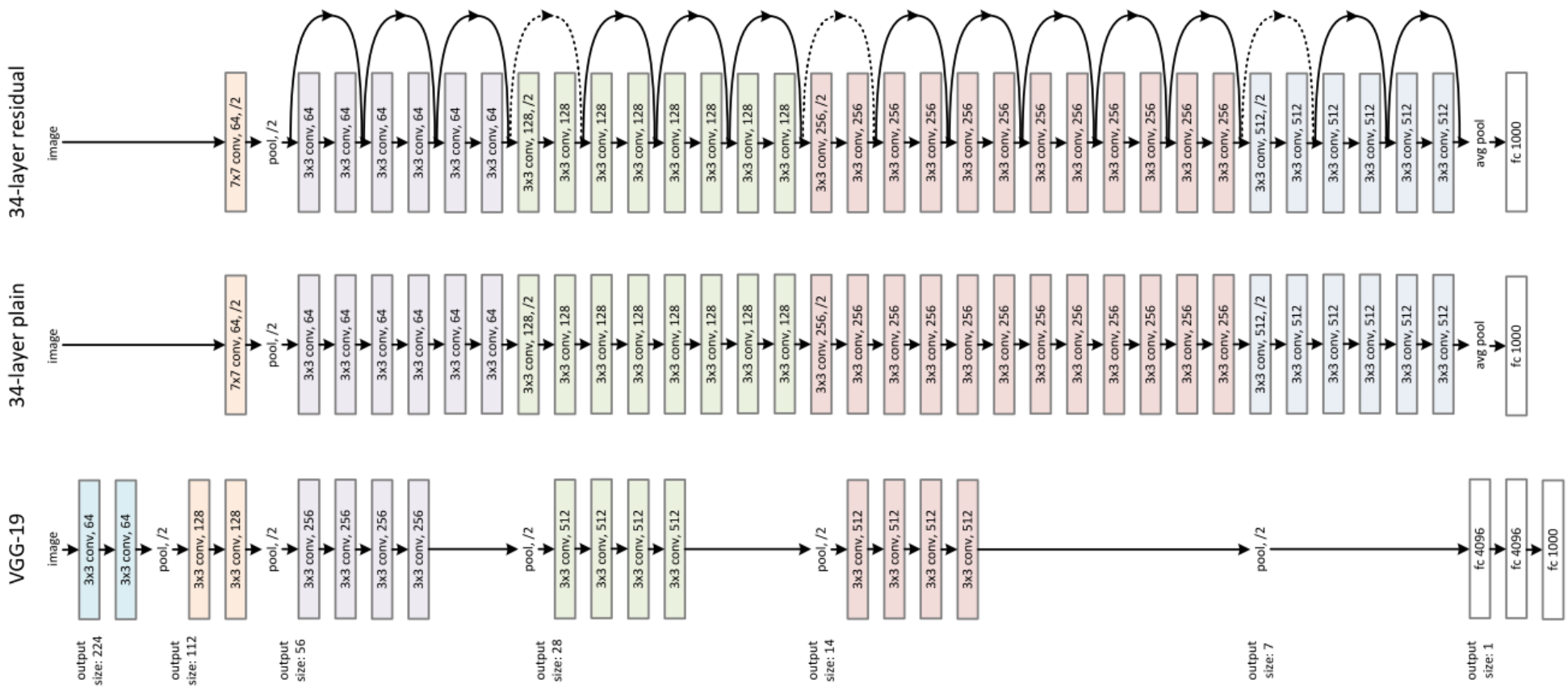
These results show that Batch Normalization should be placed before the activation functions (common mistake).

BN Experiment 2 : A Deeper CNN Model



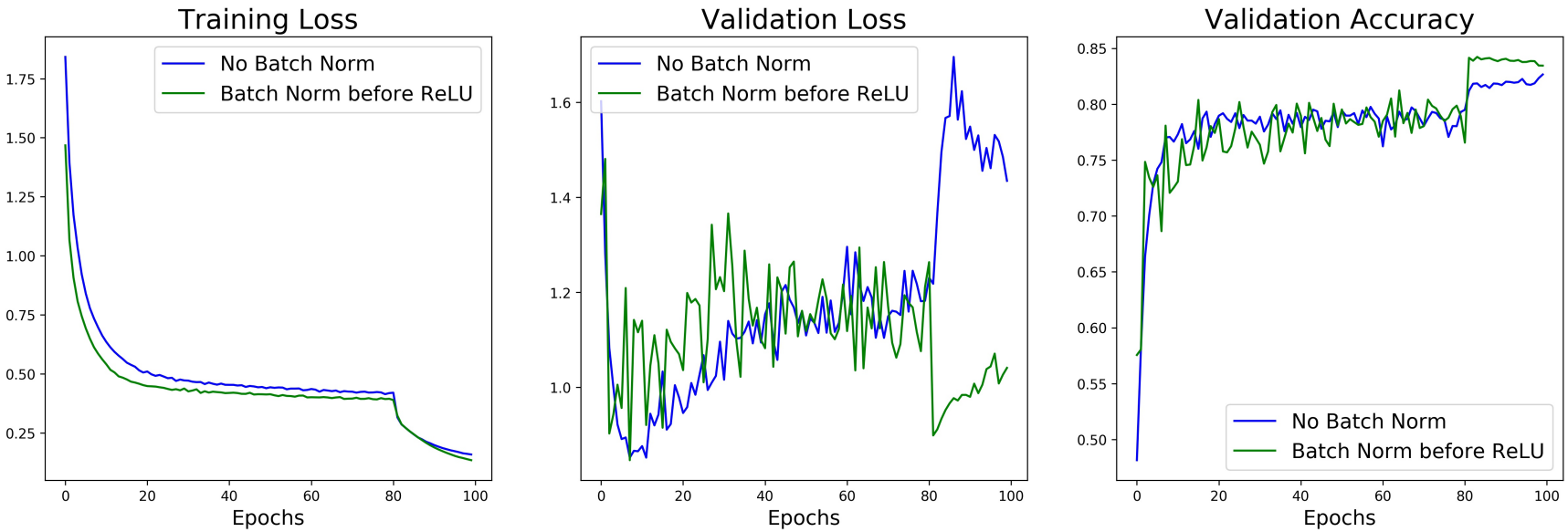
The architecture is the same as before but with twice the number of layers.

BN Experiment 3 : A ResNet20 Model



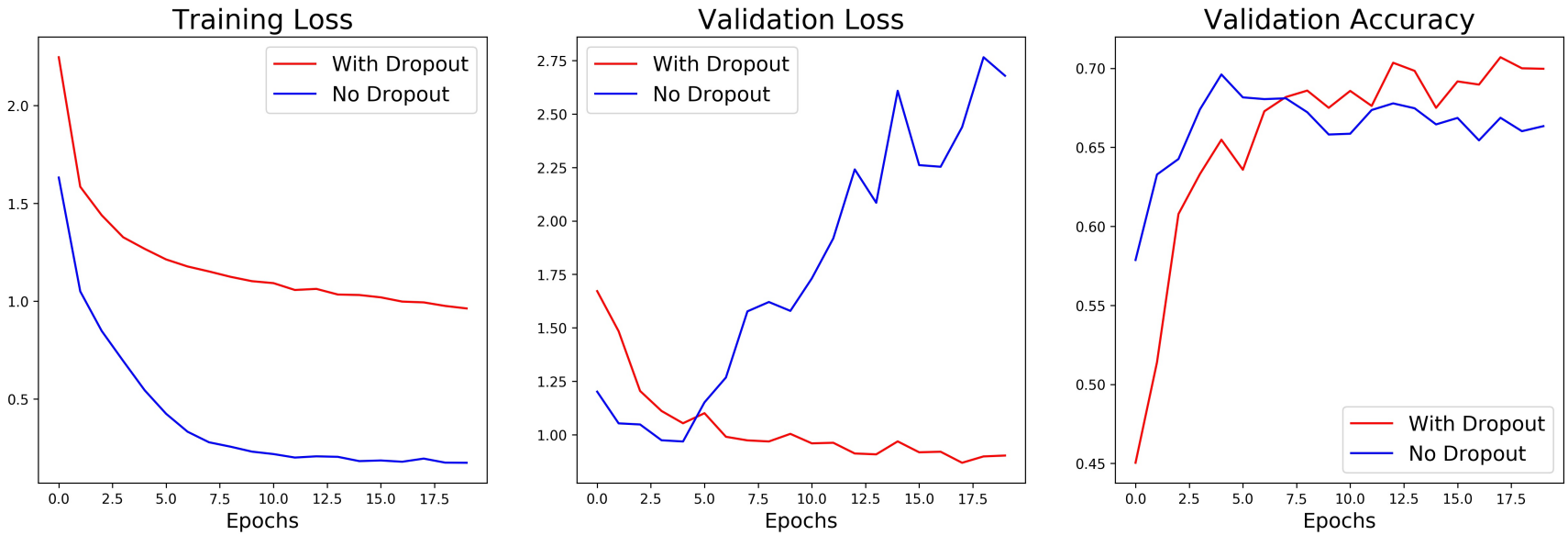
ResNets are extremely deep models that take notoriously many epochs to train but achieve state-of-the-art results.

BN Experiment 3 : A ResNet20 Model



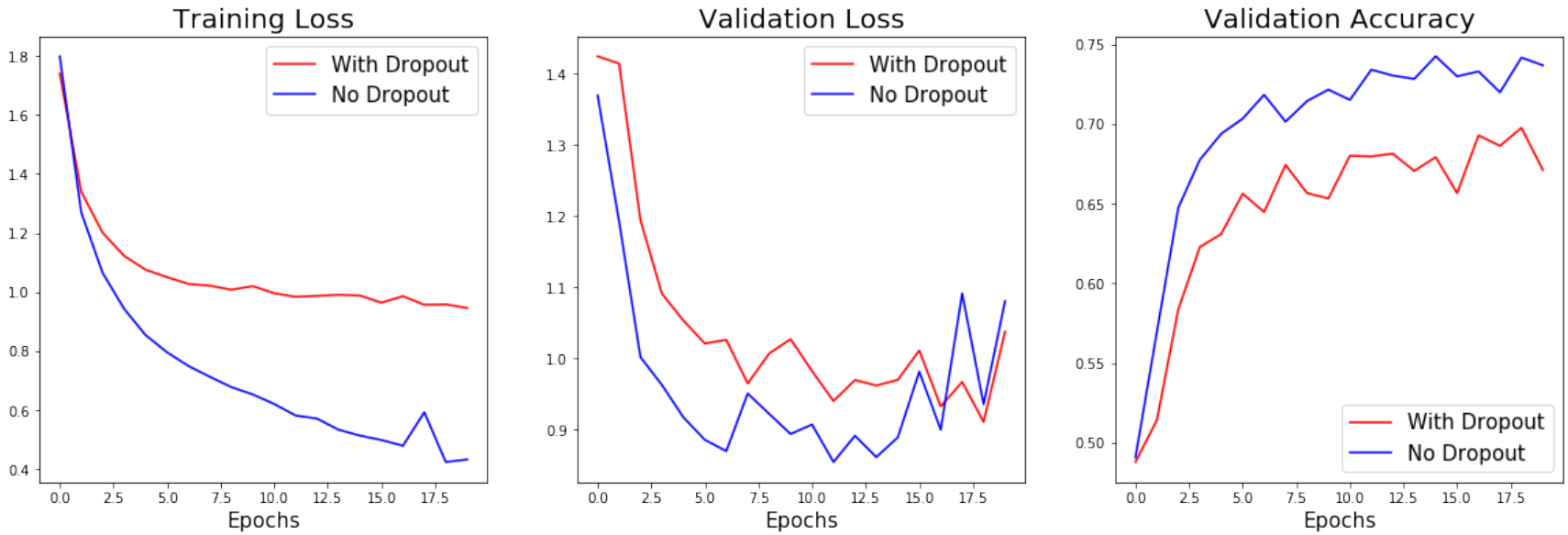
The benefit of Batch Normalization is still there but it is not as dramatic as in the previous experiments.

Dropout Experiment 1 : A simple CNN Model



Even though the training loss is higher when applying Dropout, the validation loss and accuracy are better thus highlighting its powerful regularisation effect.

Dropout Experiment 2 : A Deeper CNN Model



Dropout can induce underfitting if the probability of dropping a connection is too high. Therefore, this parameter should be tuned carefully.

Conclusion

- Batch Normalization is very powerful to speed-up training and regularize the network.
- Experiments suggest it can be universally applied with next to downside or parameter tuning.
- Dropout efficiently regularizes the network but each Dropout operation adds a dimension of hyperparameter tuning. It should be applied carefully.