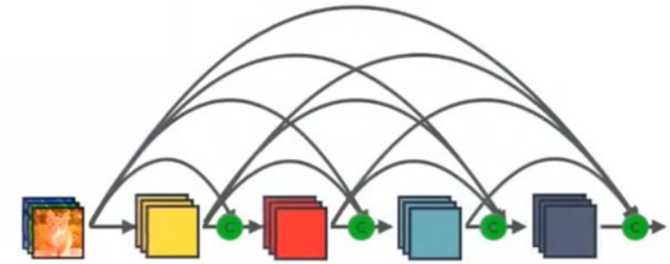
NC STATE UNIVERSITY

Dense2Net:

A Hybrid DenseNet/Res2Net Architecture



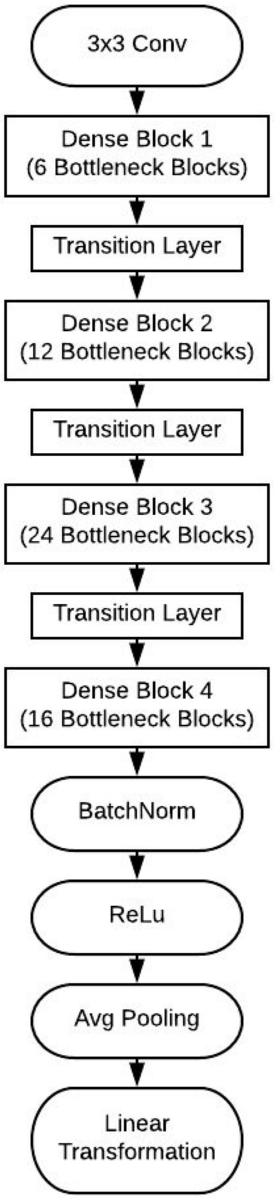
Nathan Starliper

Tanner Songkakul

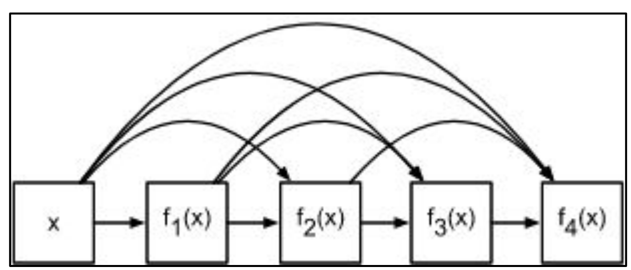
ECE763- Spring 2019

Dense2Net

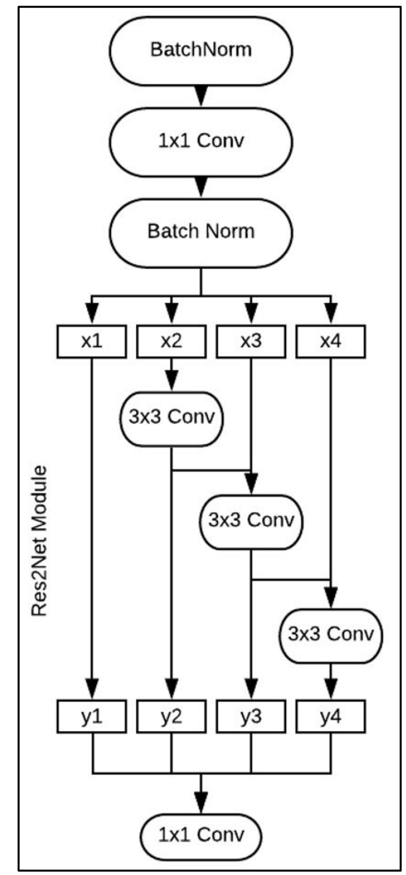
Goal: Improve classification performance by integrating Res2Net architecture into DenseNet structure



Dense2Net121



Dense Block



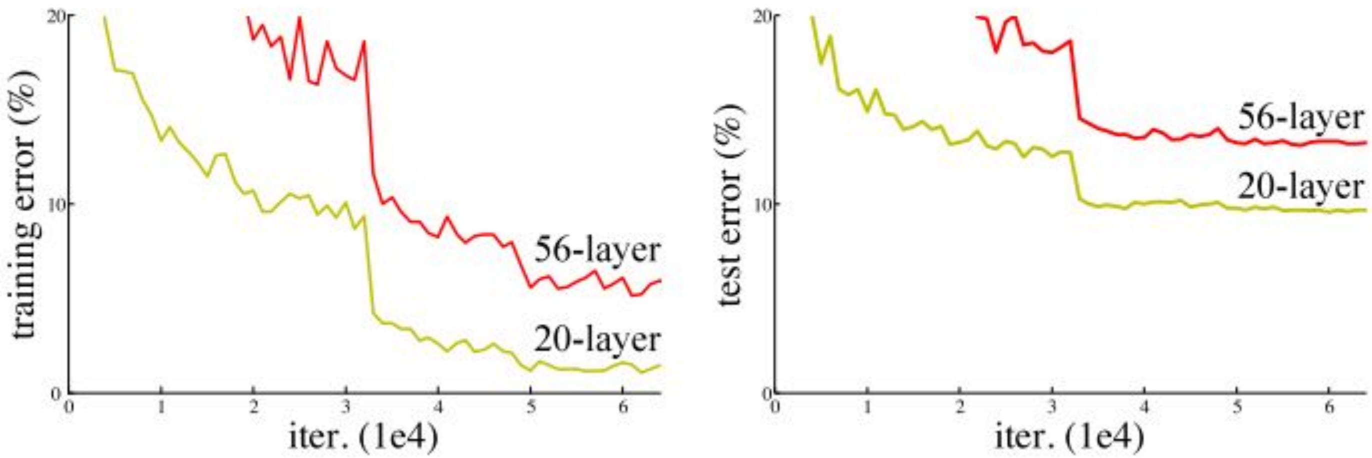
Dense2Net
Bottleneck Block

Overview

- Background
  - Residual Networks
  - DenseNet
  - Res2Net
- Dense2Net
  - Implementation
  - Training Results
  - Testing Results
  - Analysis
- Conclusions and Future Work

Background

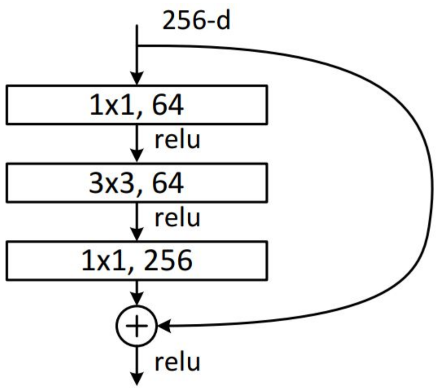
- The Problem: Vanishing Gradients
  - During backpropagation for deep networks with standard CNN architectures, gradients can become very very small
  - Increasing network depth no longer improves performance



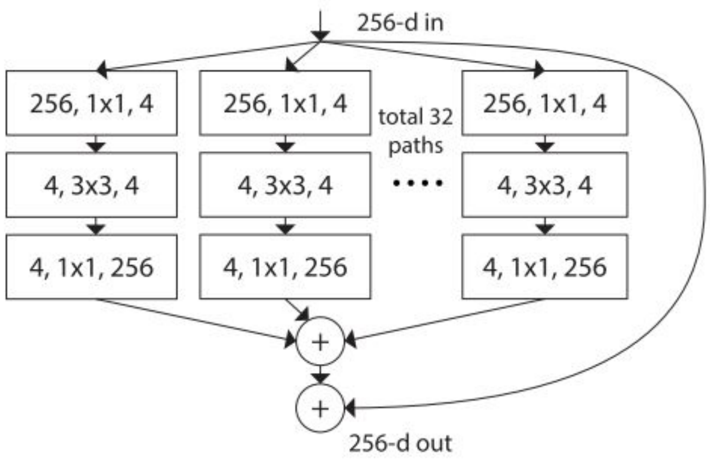
Performance comparison between 20 and 56 layer NN

Background

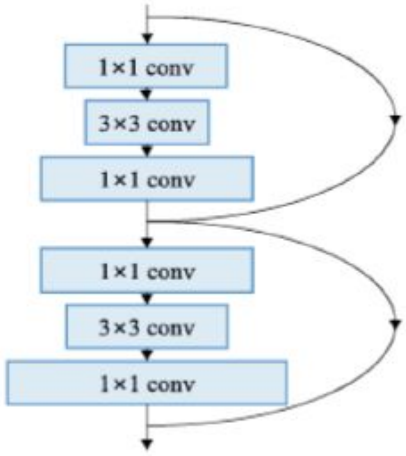
- A Solution: Residual Network Connections
  - Feed-forward networks with shortcut connections
  - Allows for deeper networks with improved performance



ResNet Bottleneck Block
(He et al, 2015)
1st Place, ILSVRC 2015



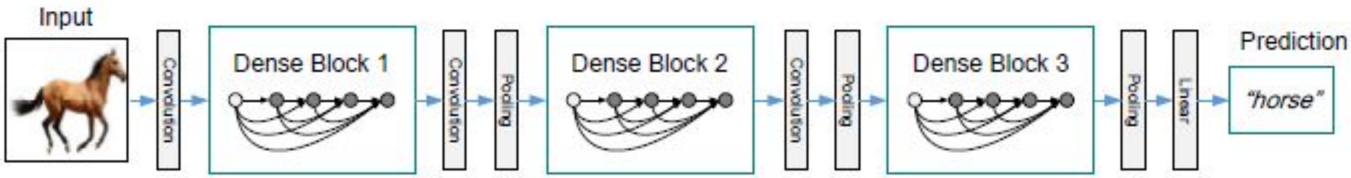
ResNext Bottleneck Blocks
(Xie et al, 2016)
2nd Place, ILSVRC 2016



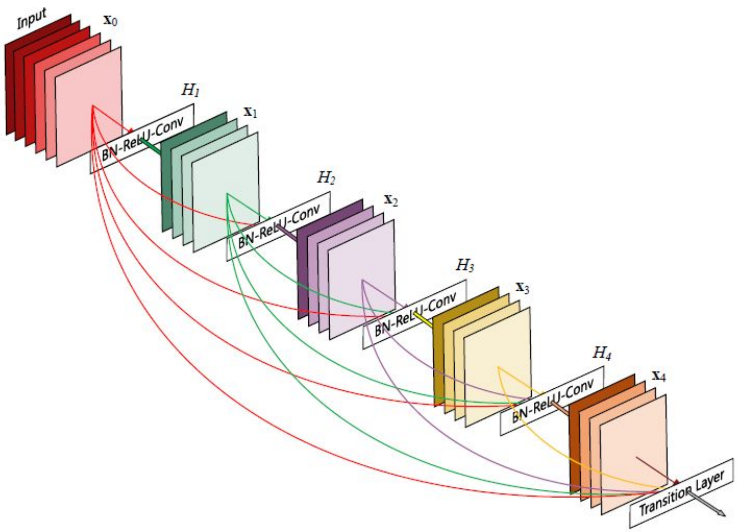
Pyramidal ResNet Bottleneck Block
(Han et al, 2017)

DenseNet

- In DenseNet (Huang et al, 2016), every layer is connected to every preceding layer inside a Dense Block



A full DenseNet

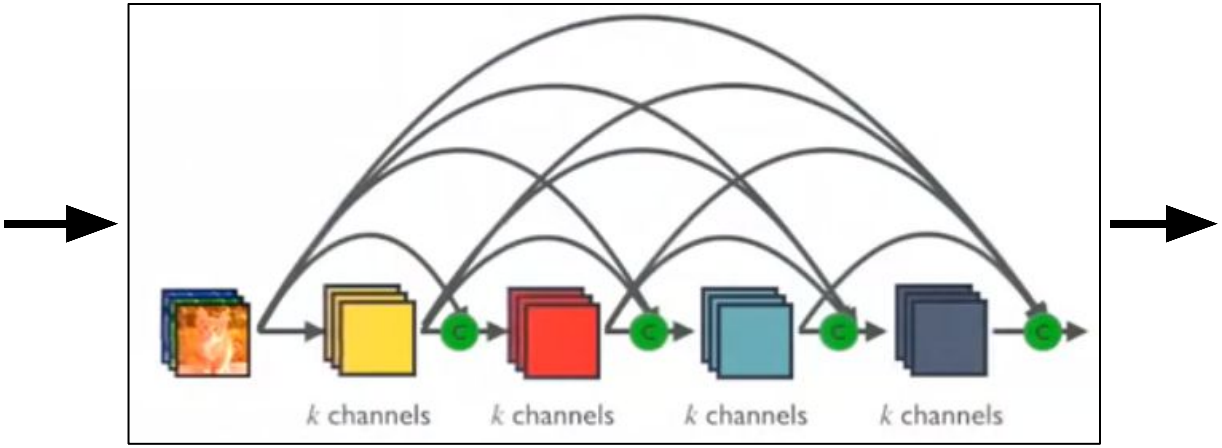


Connections within a DenseBlock

DenseNet

- Concatenates filtered outputs and skip connections:

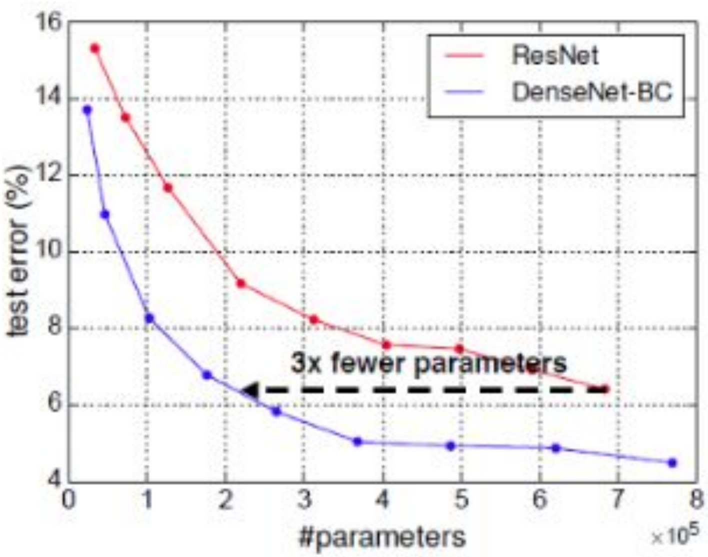
$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

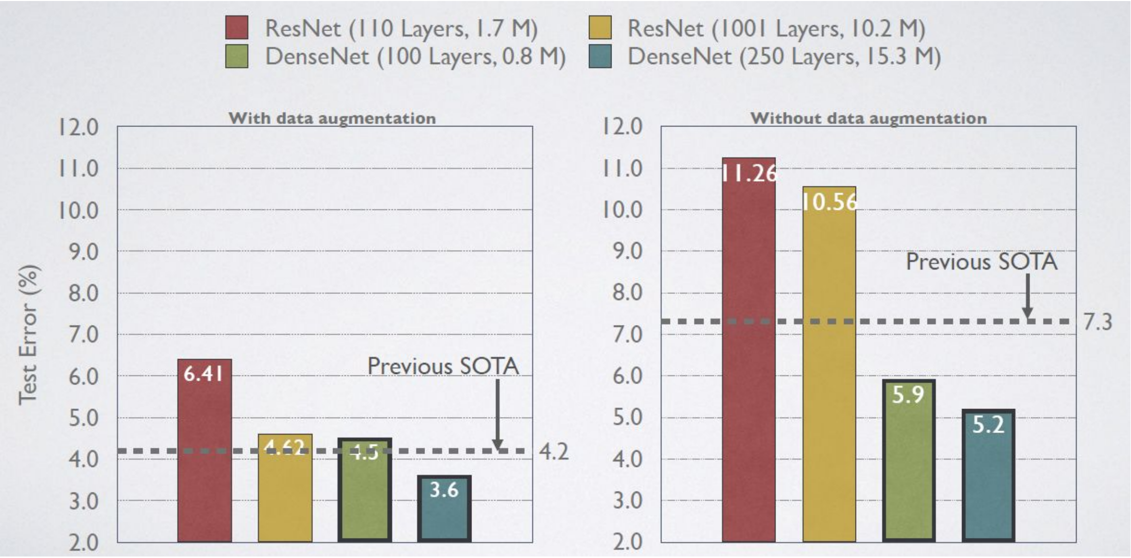


- **Dense Blocks** comprised of multiple **Dense Layers**
- Channels per Dense Layer defined by **growth rate** *k*
- **Transition Layers** between Dense Blocks downsample feature maps for concatenation
- Variation: DenseNet-BC contains **bottleneck** blocks in Dense Layers and **compression** in Transition Layers

DenseNet

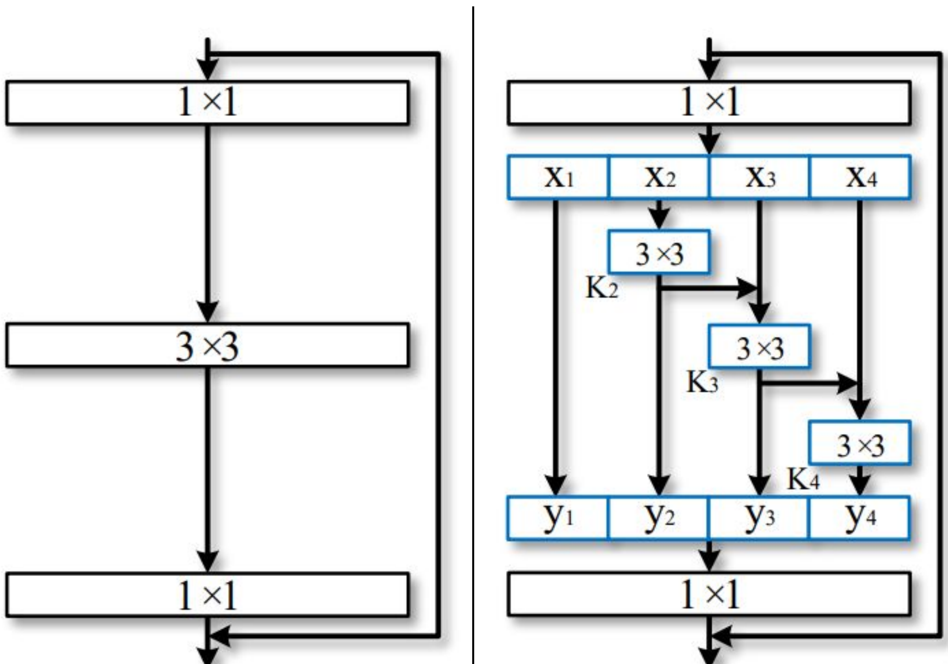
- Advantages:
  - **Gradient flow**: since every layer is directly connected, the error signal is directly propagated to earlier layers (implicit deep supervision)
  - **Efficiency**: requires less parameters per layer (proportional *l*k*k*) than ResNet
  - **Diversified Features**: every layer receives all preceding layers as input, so the features are more diverse and of all complexity levels





Res2Net

- Res2Net (Gao et al, 2019) developed to improved multi-scale feature extraction
- Replace 3x3 convolutions with a hierarchical residual network structure for increased multi-scale resolution



(a) Bottleneck block (b) Res2Net module

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i & i = 1; \\ \mathbf{K}_i(\mathbf{x}_i + \mathbf{y}_{i-1}) & 1 < i \leqslant s \end{cases}$$

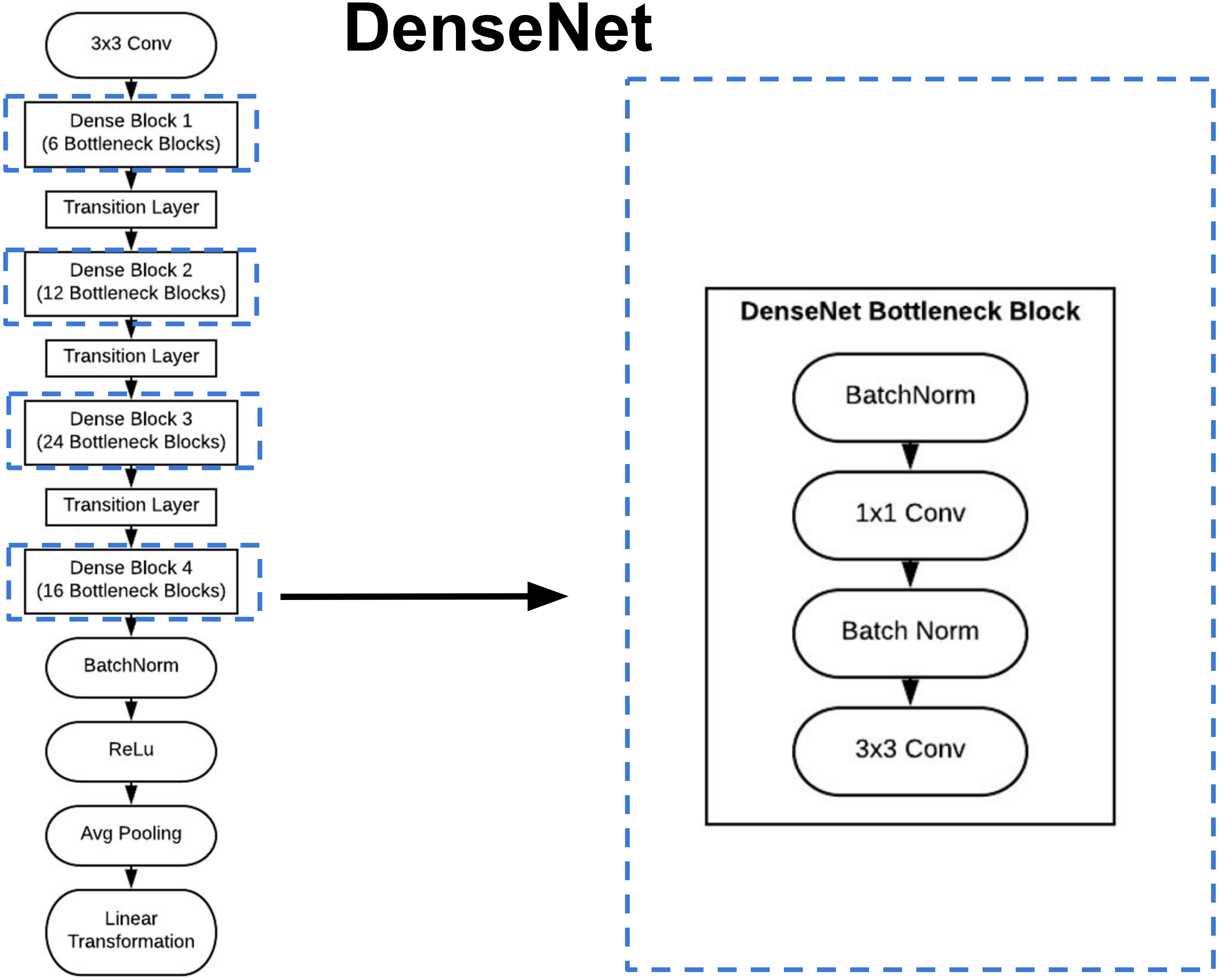
s: **scale**, number of subsets to split input into

Res2Net

- Scale *s* is orthogonal to other characteristics of NN
- Bottleneck blocks in many networks can be changed to Res2Net architecture (Res2Net, Res2Next, Res2Next-DLA) to improve performance
- Hasn't been integrated with DenseNet

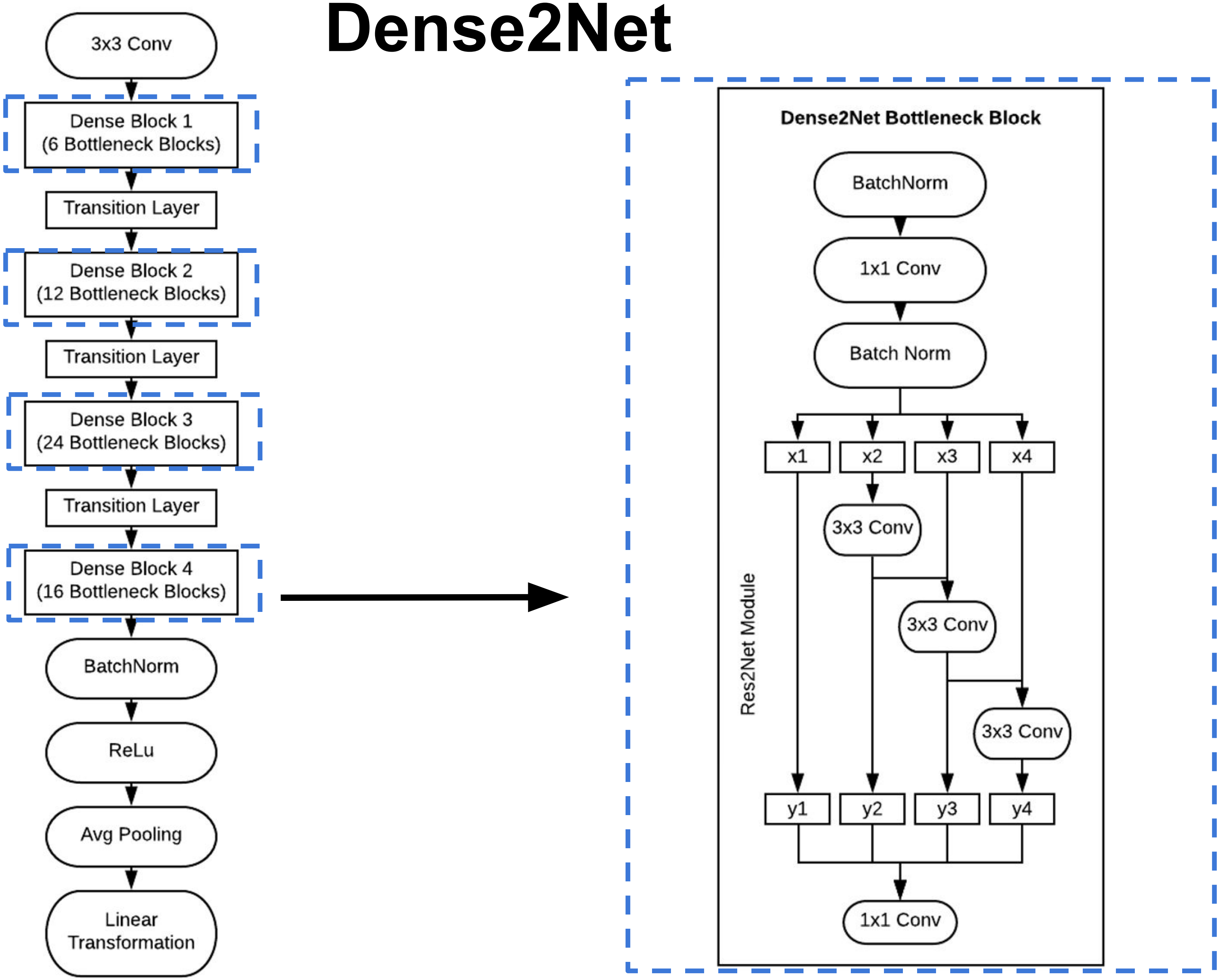
	Params	top-1 err.
Wide ResNet [48]	36.5M	20.50
ResNeXt-29, 8c×64w [43] (base)	34.4M	17.90
ResNeXt-29, 16c×64w [43]	68.1M	17.31
DenseNet-BC (k = 40) [20]	25.6M	17.18
Res2NeXt-29, 6c×24w×4scale	24.3M	16.98
Res2NeXt-29, 8c×25w×4scale	33.8M	16.93
Res2NeXt-29, 6c×24w×6scale	36.7M	**16.79**
ResNeXt-29, 8c×64w-SE [19]	35.1M	16.77
Res2NeXt-29, 6c×24w×4scale-SE	26.0M	16.68
Res2NeXt-29, 8c×25w×4scale-SE	34.0M	16.64
Res2NeXt-29, 6c×24w×6scale-SE	36.9M	**16.56**

DenseNet



Replace 3x3 conv with Res2Net module+1x1 conv

Dense2Net



Replace 3x3 conv with Res2Net module+1x1 conv

Implementation

- Dense2Net implemented in PyTorch
- 4 Dense Blocks: 6 layers→12 layers→24 layers→16 layers, Res2Net incorporated into each layer
- SGD optimizer: weight decay 0.0001, momentum 0.9, batch size 128, 100 epochs

```
class Bottleneck(nn.Module):
  #Dense2net bottleneck block
    def __init__(self, in_planes, growth_rate):
        global dense2net
        super(Bottleneck, self).__init__()
        self.bn1 = nn.BatchNorm2d(in_planes)
        self.conv1 = nn.Conv2d(in_planes, 4*growth_rate, kernel_size=1, bias=False)
        self.bn2 = nn.BatchNorm2d(4*growth_rate)
        if dense2net:
          self.conv2 = Res2Net_block(4*growth_rate, scale=4, stride=1, groups=1)
          self.conv3 =nn.Conv2d(4*growth_rate, growth_rate, kernel_size=1, bias=False)
        else:
          self.conv2 = nn.Conv2d(4*growth_rate, growth_rate, kernel_size=3, padding=1, bias=False)

    def forward(self, x):
        global dense2net
        out = self.conv1(F.relu(self.bn1(x)))
        out = self.conv2(F.relu(self.bn2(out)))
        if dense2net:
          out = self.conv3(out)
        out = torch.cat([out,x], 1)
        return out
```

Experiments

- CIFAR-100 dataset, 100 epochs training
- Backbone architecture: DenseNet121BC with k=32
- Multiple configurations tested
  - Data Augmentation, cutout, scale, cardinality, and Squeeze and Excitation
- Efficient training
  - Learning rate decay on plateau
  - Early stopping

Initial Baseline Results

LR Decay by factor of 10

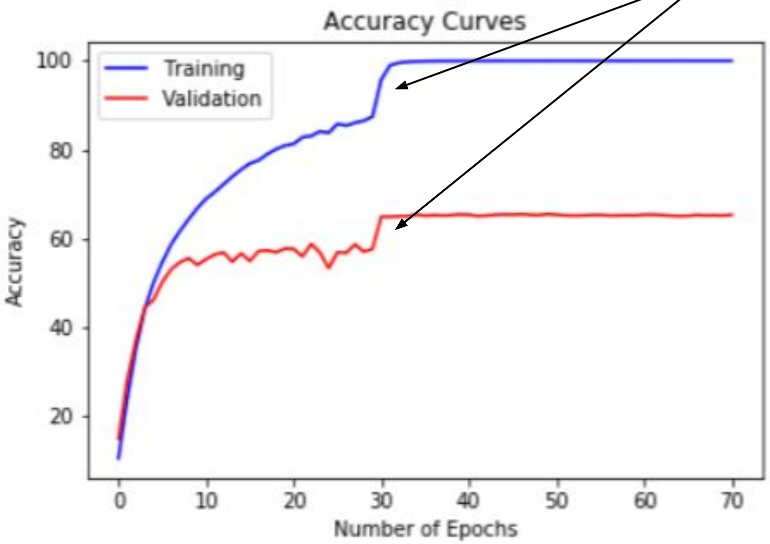


Figure 8: Accuracy curves of the baseline DenseNet model without data augmentation that we will compare our implementations against.

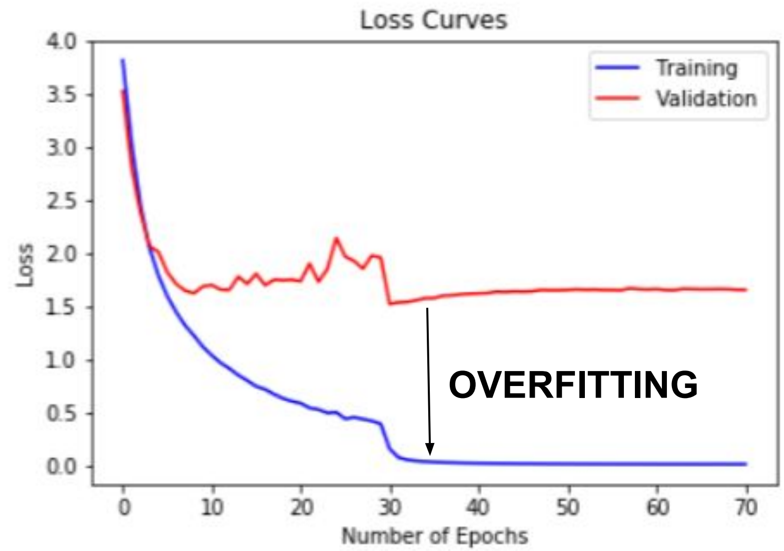
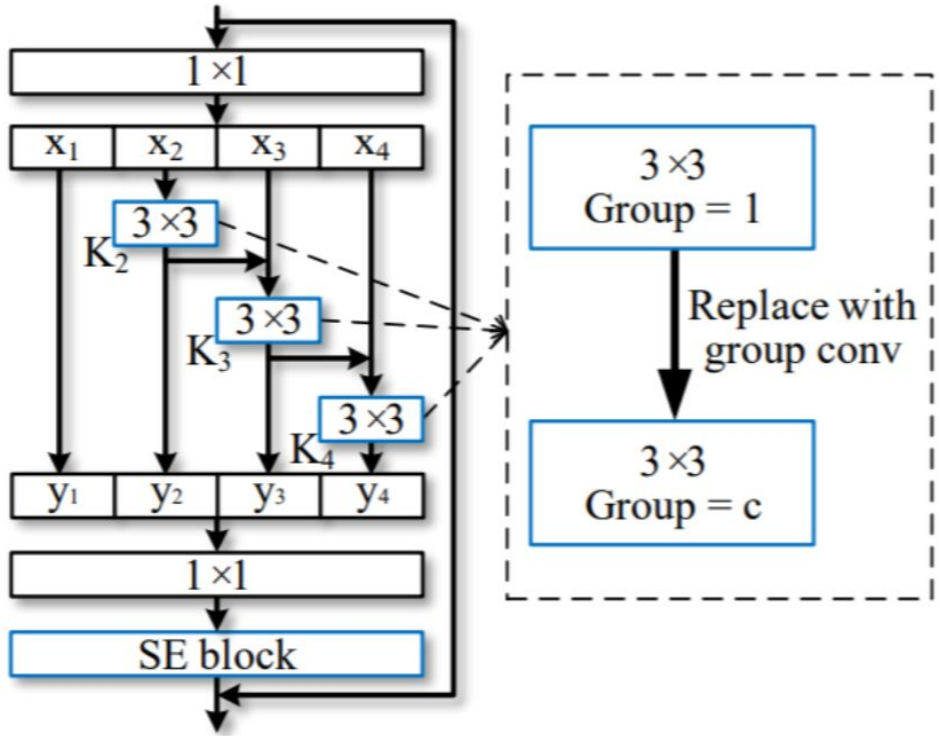


Figure 9: Loss curves of the baseline DenseNet model without data augmentation that we will compare our implementation against.

Data Augmentation/Cutout

- Standard transformations: mirroring/shifting
- Cutout: mask random patches of images during training
  - N_holes = 1
  - Length = 8
- Regularization to reduce overfitting

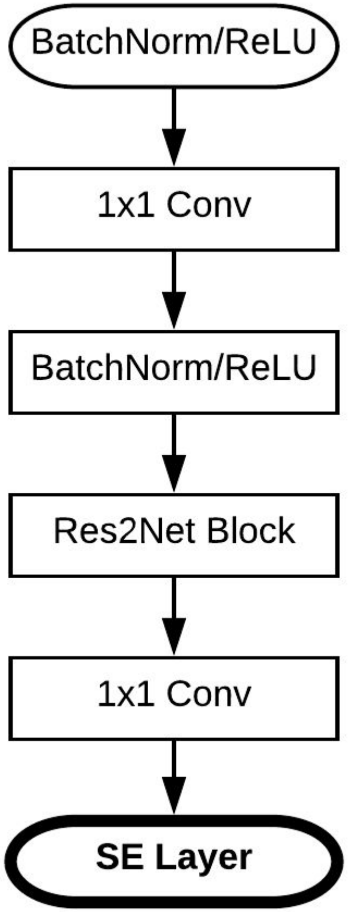
Scale and Cardinality



- **Scale**: increases granularity and scales of extracted features
  - Scale = 2,4,8
- **Cardinality**: group convolution enforces block-diagonal structure sparsity on channel dim (induces regularization)
  - Cardinality = 32

Squeeze and Excitation Block

- SE layer weights output filters based on importance
- Emphasizes informative while suppressing uninformative features
- Feature "calibration"
- Added at the end of the bottleneck block



Experimental Results Summary

Model Config	Valid Acc (%)	Train Acc (%)
DenseNet No Augmentation	65.41	99.98
DenseNet	71.91	98.7
Dense2Net No Augmentation	68.96	99.98
Dense2Net	75.85	99.95
Dense2Net $scale = 2$	75.26	99.35
Dense2Net $scale = 8$	76.1	98.77
Dense2Net with Cutout	75.23	98.33
Dense2Net with Grouped Convs	75.96	98.67
Dense2Net with SE	74.77	98.68

- Dense2Net: 5.83% relative increase over baseline DenseNet
- Data Augmentation: 9.99% relative increase
- Minor (possibly insignificant) increases due to scale

Final Results

- Closed gap between training and validation

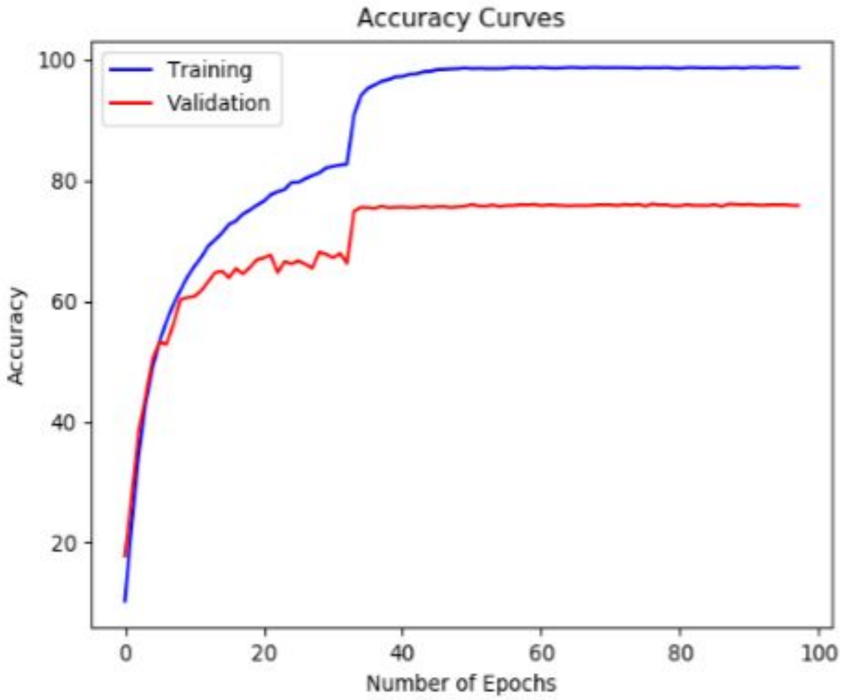


Figure 10: Accuracy curves of the best Dense2Net model with data augmentation and $scale = 8$.

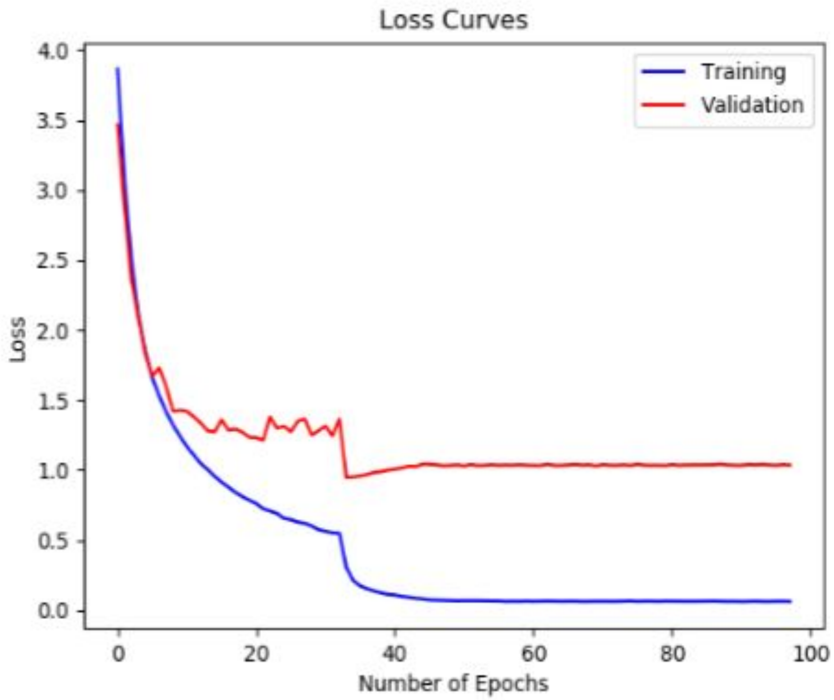


Figure 11: Loss curves of the best Dense2Net model with data augmentation and $scale = 8$.

Conclusion and Future Work

- Showed benefit of introducing the Res2Net block into DenseNet
- Rigorous fine tuning of hyperparameters will yield better results
  - Scale, cardinality, growth rate, learning rate decay, optimizer
- Implement Res2Net in other architectures
- Test on larger images that would benefit from high scale extraction (ImageNet)