SVKM'S NMIMS Deemed to be UNIVERSITY

Transfer Learning

B.Tech. Data Science, NMIMS

By,

Bilal Hungund, Data Scientist, Halliburton

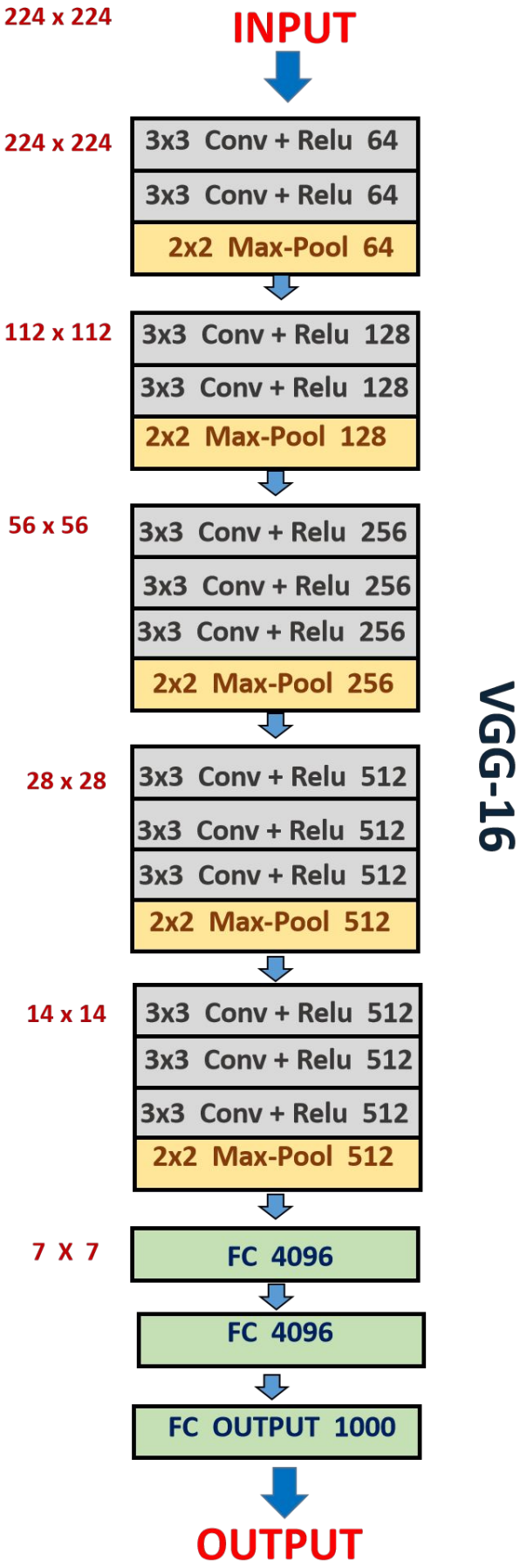
Transfer Learning

→ In 2010, Stanford created the Image Net Competition.

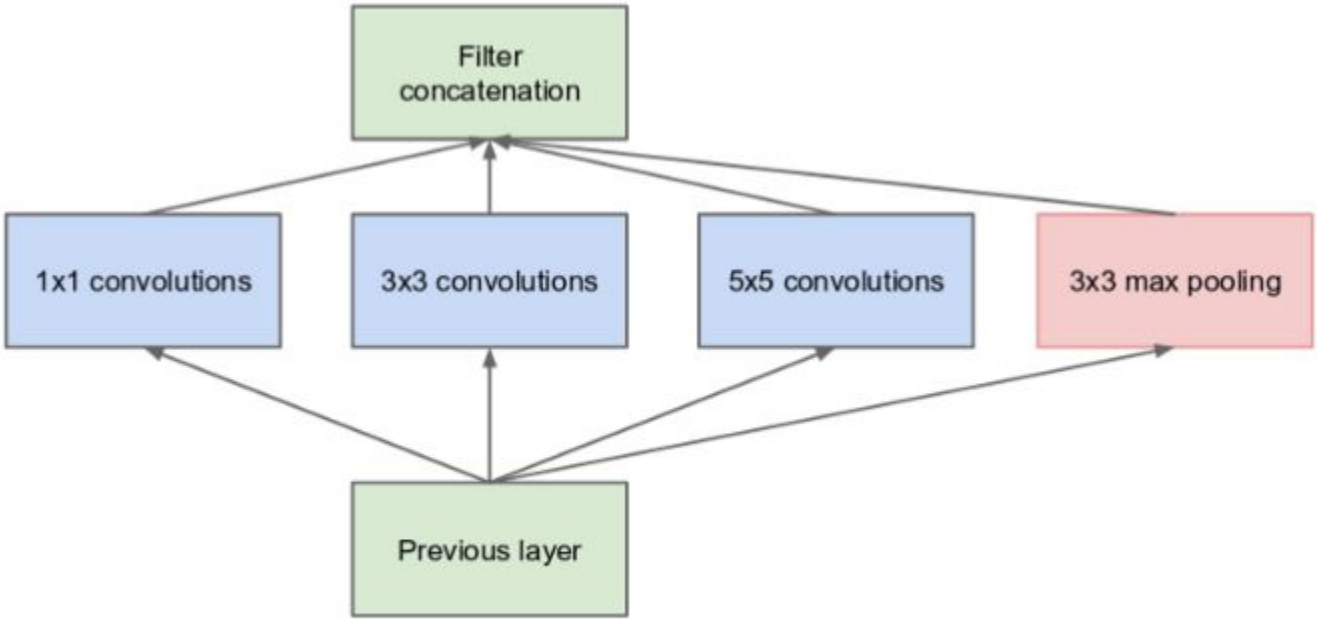
→ Over the years different types of convolutional neural network are designed to optimized the error rate.

→ Transfer Learning allows for the knowledge learned in one task to be reused as a starting point for a second task.

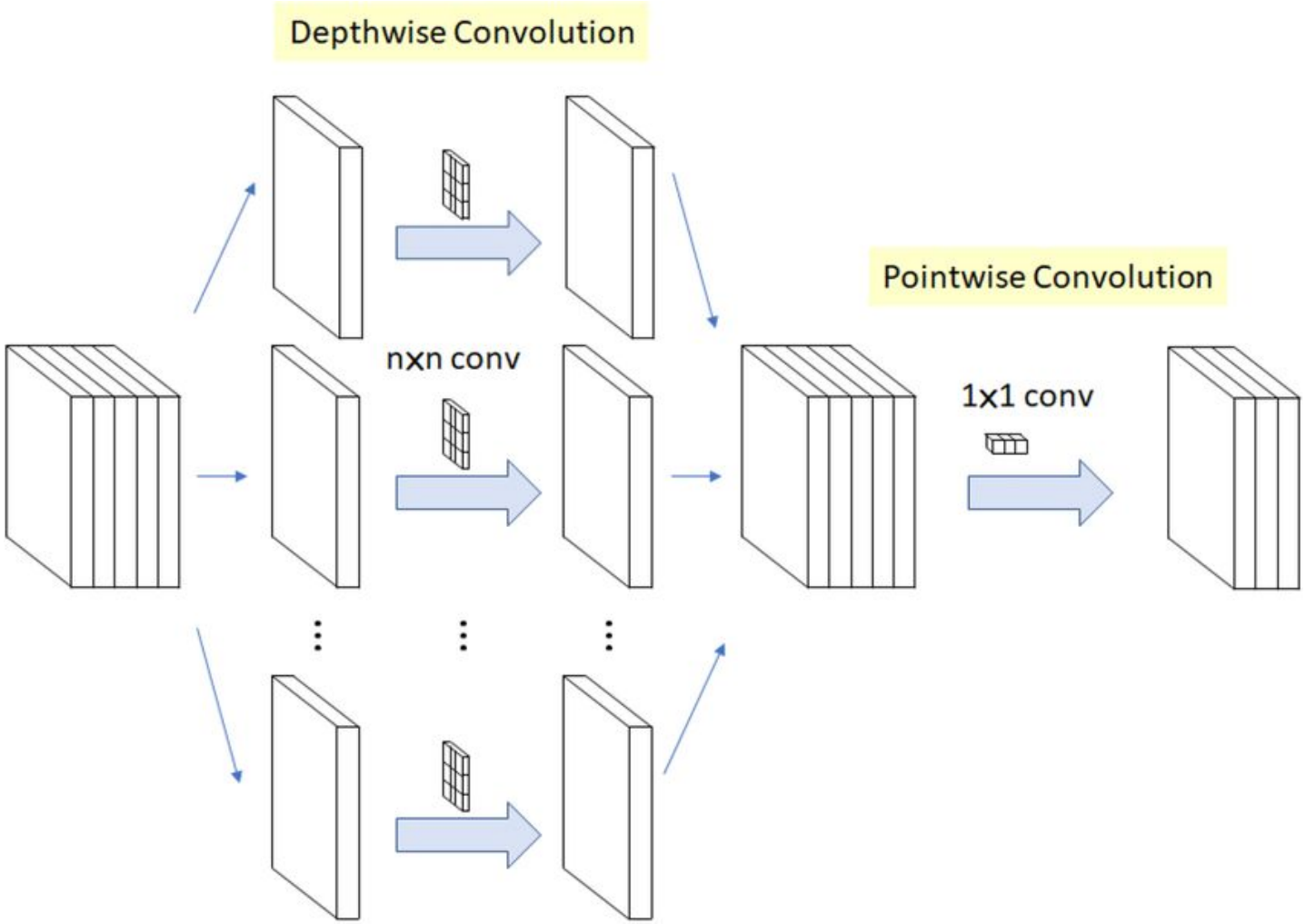
→ Example: model capable to identify humans, animals, and furniture locations can be modified to find objects from real time image data

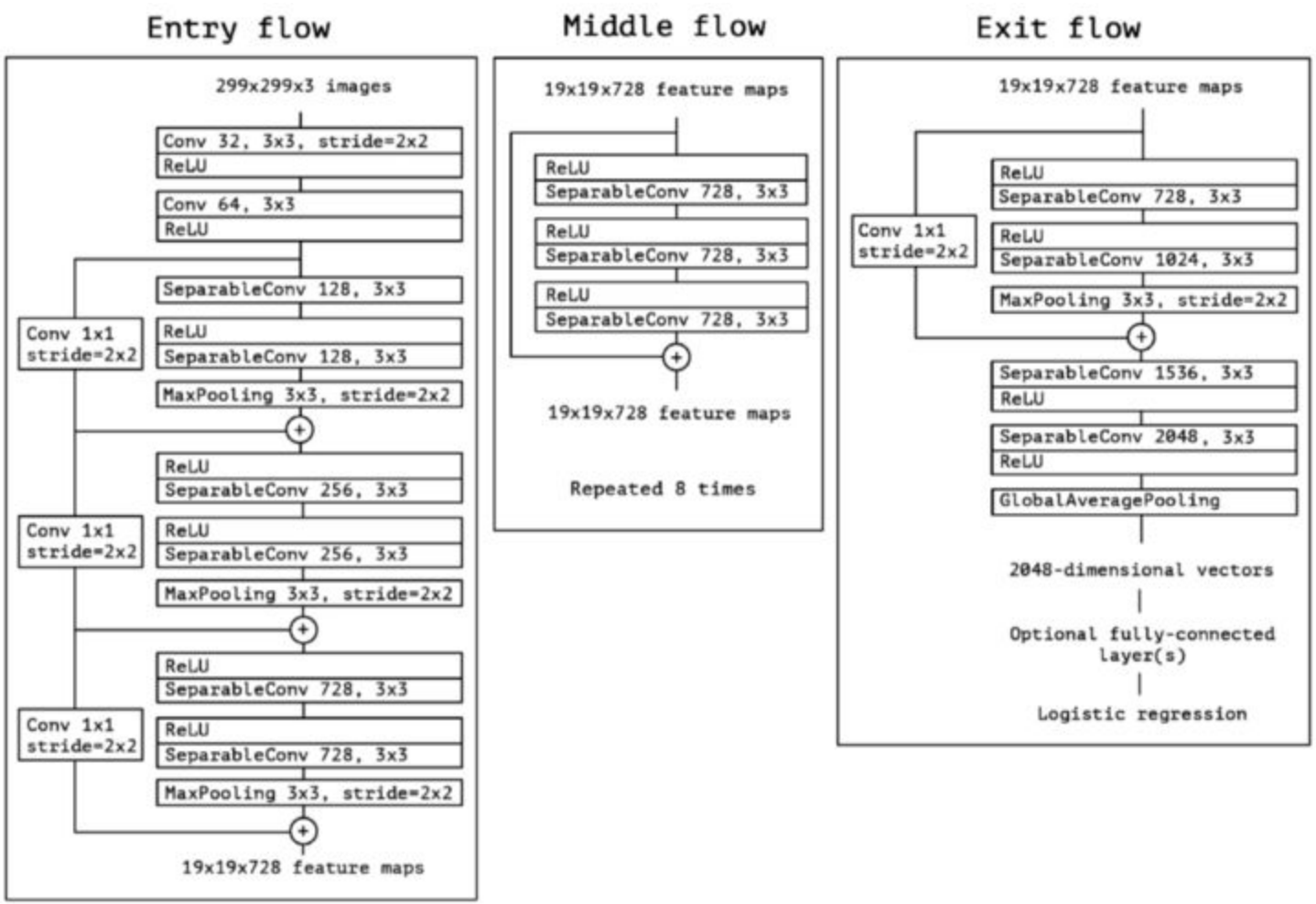

224 x 224
INPUT
224 x 224
3x3 Conv + Relu 64
3x3 Conv + Relu 64
2x2 Max-Pool 64
112 x 112
3x3 Conv + Relu 128
3x3 Conv + Relu 128
2x2 Max-Pool 128
56 x 56
3x3 Conv + Relu 256
3x3 Conv + Relu 256
3x3 Conv + Relu 256
2x2 Max-Pool 256
28 x 28
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
14 x 14
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
7 X 7
FC 4096
FC 4096
FC OUTPUT 1000
OUTPUT
VGG-16

Inception Models



Imran Juneja




Entry flow
299x299x3 images
Conv 32, 3x3, stride=2x2
ReLU
Conv 64, 3x3
ReLU
SeparableConv 128, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 128, 3x3
MaxPooling 3x3, stride=2x2
ReLU
SeparableConv 256, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 256, 3x3
MaxPooling 3x3, stride=2x2
ReLU
SeparableConv 728, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 728, 3x3
MaxPooling 3x3, stride=2x2
19x19x728 feature maps
Middle flow
19x19x728 feature maps
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 728, 3x3
19x19x728 feature maps
Repeated 8 times
Exit flow
19x19x728 feature maps
Conv 1x1 stride=2x2
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 1024, 3x3
MaxPooling 3x3, stride=2x2
SeparableConv 1536, 3x3
ReLU
SeparableConv 2048, 3x3
ReLU
GlobalAveragePooling
2048-dimensional vectors
Optional fully-connected layer(s)
Logistic regression

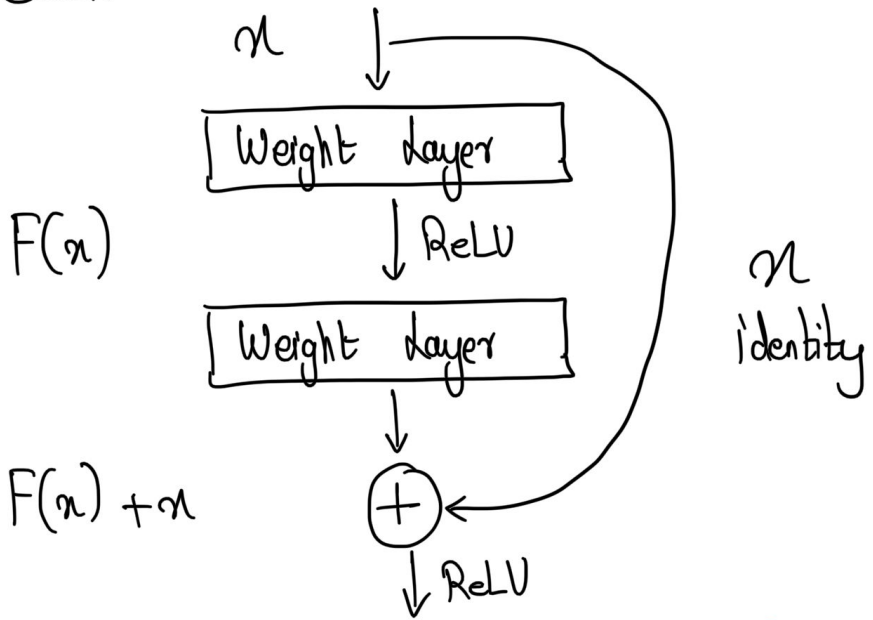
Residual Network (Resnet)

→ Adding convolution layers deeper leads to vanishing gradients problem and it impact the model performance

→ Vanishing gradients can be easily blame to overfit the model, though the authors argue that the use of Batch Normalization ensures that the gradients have healthy norms.

→ The problem of training very deep networks has been alleviated with the introduction of a new neural network layer - The Residual Block

Residual Block

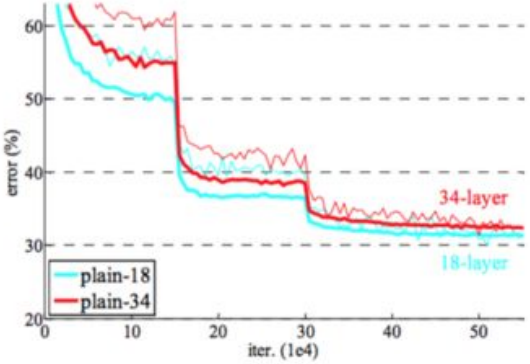


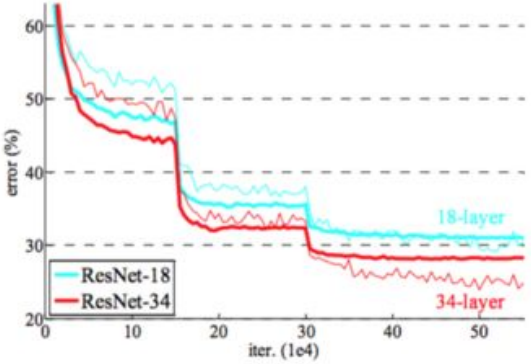
$$y = F(x, W_i) + W_s x$$

$W_s x$ → Square weight matrix

$F(x, W_i)$ → Residual Mapping to be learned

https://arxiv.org/abs/1512.03385





	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

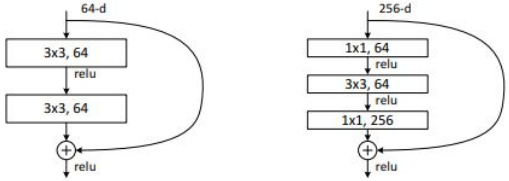
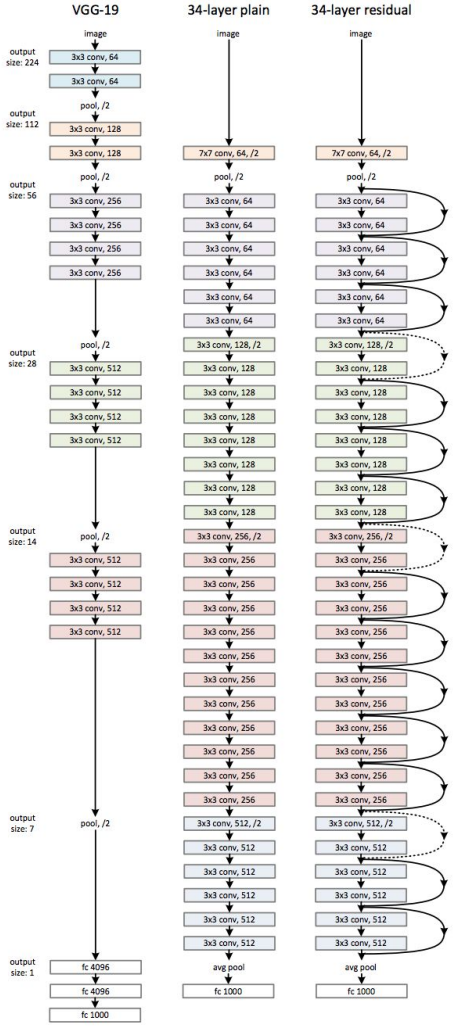


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

https://arxiv.org/abs/1512.03385



ResNet Variants

- ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202
- The name ResNet followed by a two or more digit number simply implies the ResNet architecture with a certain number of neural network layers.
- ResNet1202 overfits.
- Batch Normalization used after each convolution and before activation.
- Do not use Dropout.
- In Conclusion, the Skip Connection is a very interesting extension to Deep Convolutional Networks that have empirically shown to increase performance in ImageNet classification

DenseNets

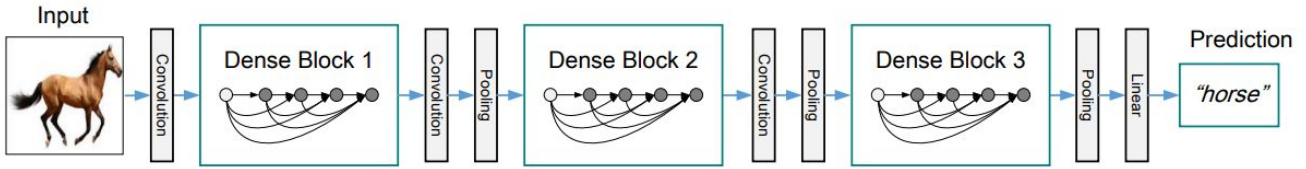
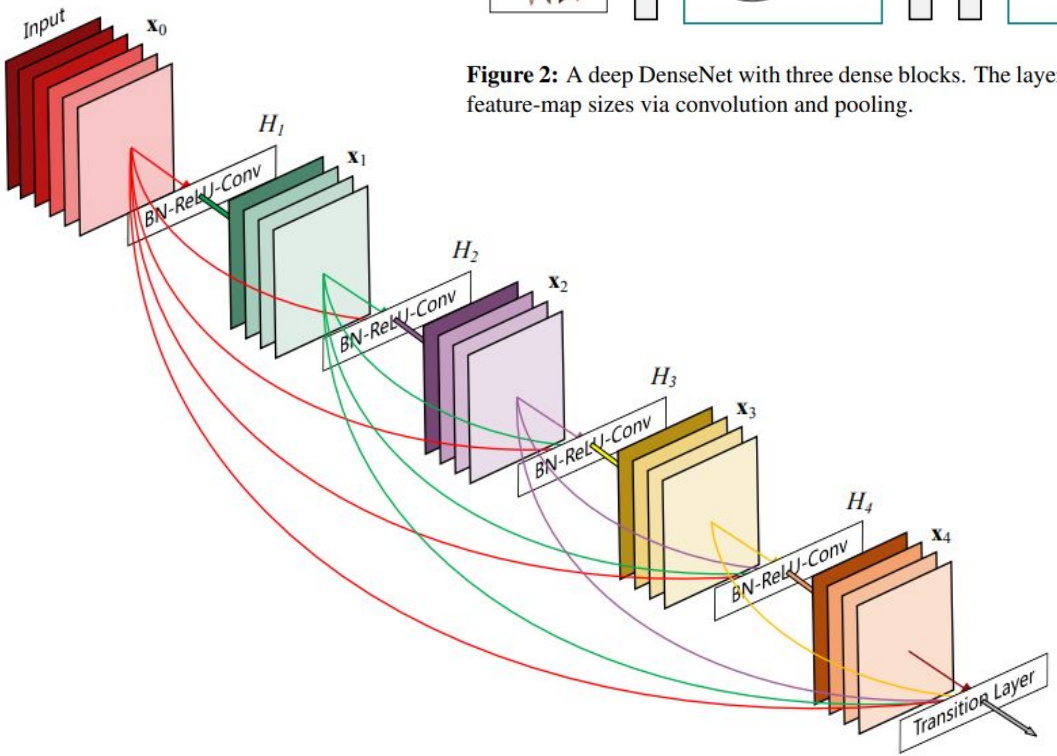
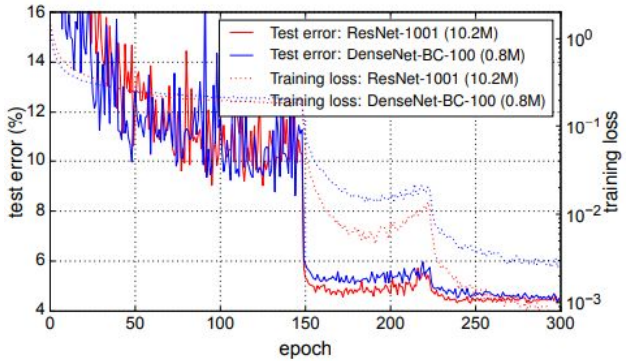


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

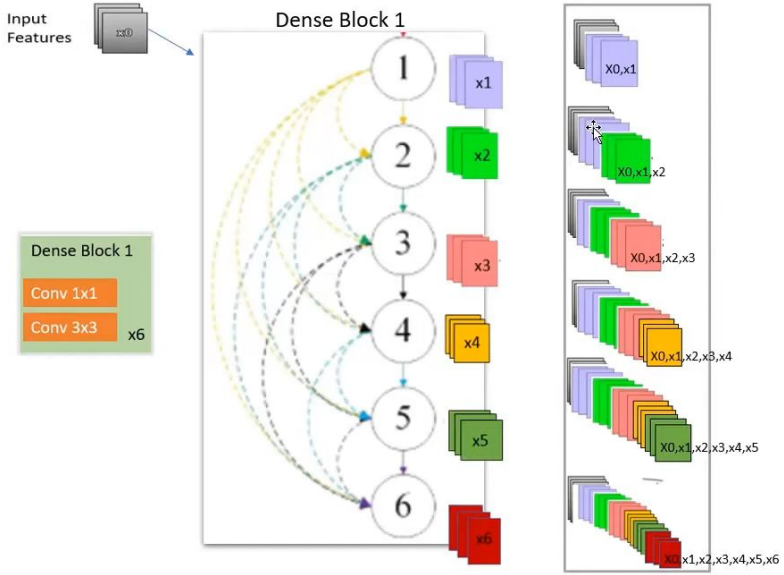




https://arxiv.org/pdf/1608.06993.pdf

Inside each Dense Block and DenseNet Variants

- densenet121, densenet169, densenet201, densenet161



Fastai Models

The fastai library includes several pretrained models from torchvision, namely:

- resnet18, resnet34, resnet50, resnet101, resnet152
- squeezenet1_0, squeezenet1_1
- densenet121, densenet169, densenet201, densenet161
- vgg16_bn, vgg19_bn
- alexnet

References

- https://arxiv.org/pdf/1512.03385.pdf
- https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4
- https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338
- https://arxiv.org/abs/1608.06993
- https://youtu.be/hCg9bolMeJM
- https://fastai1.fast.ai/vision.models.html