Histopathologic Cancer Detection

DSCI 419 FINAL PROJECT

STUDENT A

2018

Agenda

Project Overview

Method

- Issues with Deep Network
- Residual Neural Network (ResNet)
- Densely Connected Neural Network (DenseNet)
- Transfer Learning

Modeling and Results

- Baseline
- ResNet50
- DenseNet 121

Conclusion

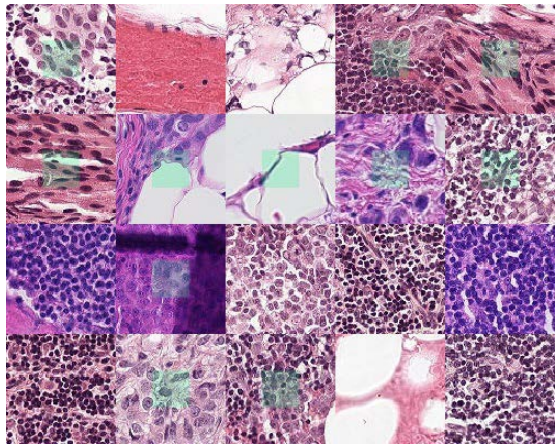
Histopathologic Cancer Detection

Objective

- To identify metastatic cancer in small image patches taken from larger digital pathology scans.

Input Data

- Slightly modified version of the PatchCamelyon (PCam) benchmark dataset
- 96 * 96 RGB image
- Around 220,000 images (130, 000 no-tumor and 90,000 tumor)



Method

- Binary classification using CNN
  - Residual Neural Network (ResNet)
  - Densely Connected Network (DenseNet)

Model Evaluation

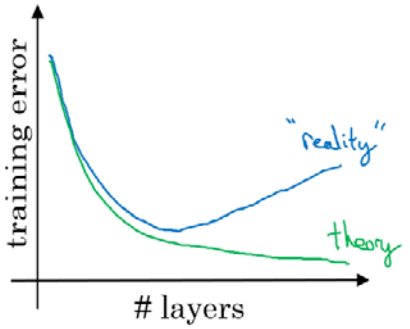
- Validation Accuracy

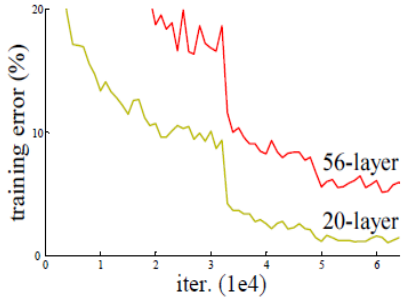
Method

Issues with Deep Neural Networks

- Vanishing/Exploding gradients
  - Normal initialization
  - Intermediate normalization layer – Batch Normalization
    - Accelerate training of deep networks

- Training accuracy degradation
  - Reality is different from theory.
  - With depth increasing, training accuracy gets saturated and then degrades rapidly.
  - Deep layers lose representation of previous layers.
  - Solutions: Residual Neural Network and Densely Connected Network



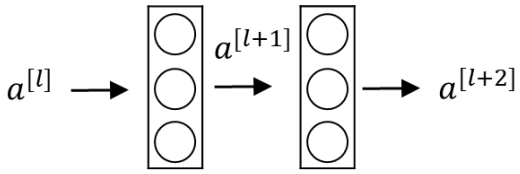


Method (Cont.)

Residual Neural Network (ResNet)

- Advanced CNN architecture
- Developed by Kaiming He, et al. (2015)
- Residual learning block
  - Add residual connection
- Improvement
  - Residual representation
    - Simplify optimization
  - Shortcut connection

Plain



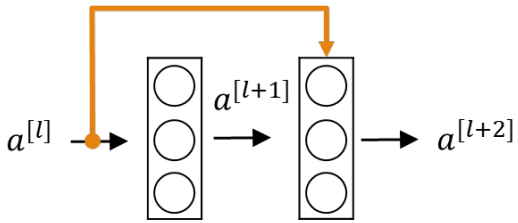
$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]}$$

$$a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = g(z^{[l+2]})$$

Residual



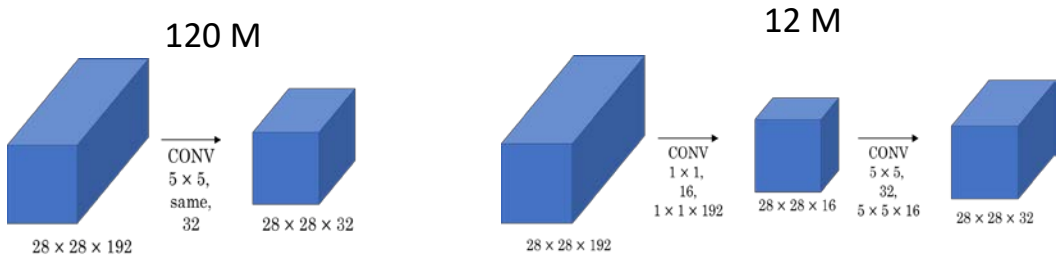
$$a^{[l+1]} = g(z^{[l+1]})$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

Method (Cont.)

ResNet50

- 50 layers with trainable parameters
- 1 by 1 Conv Layer – Bottleneck layer
  - Deeper architecture
  - Reduce the size of parameters



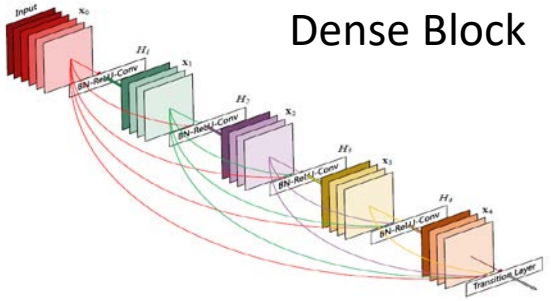
- Keras Implementation
  - keras.applications.resnet50.ResNet50

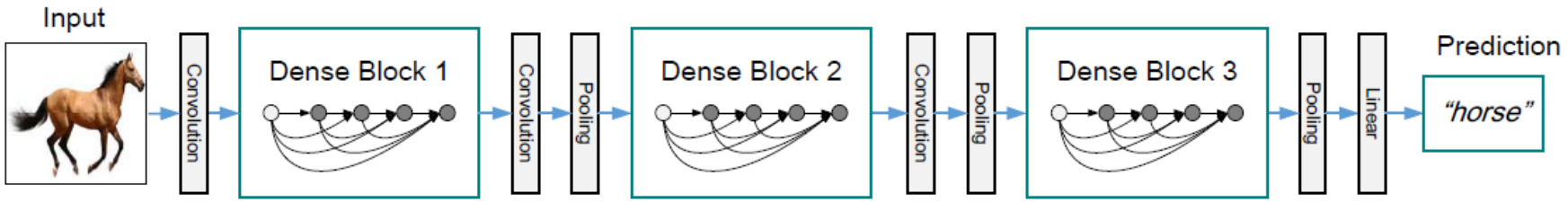
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Method (Cont.)

Densely Connected Network (DenseNet)

- Advanced CNN Architecture
- Create short paths from early layers to later layers
  - Feature reuse – easy to train and highly parameter efficient
- Address vanish or “wash out” issues, as CNNs become increasingly deep
- Layers are very narrow (e.g., 12 filters per layer)
- Final classifier makes decision based on all feature-maps in the network.





Method (Cont.)

DenseNet121

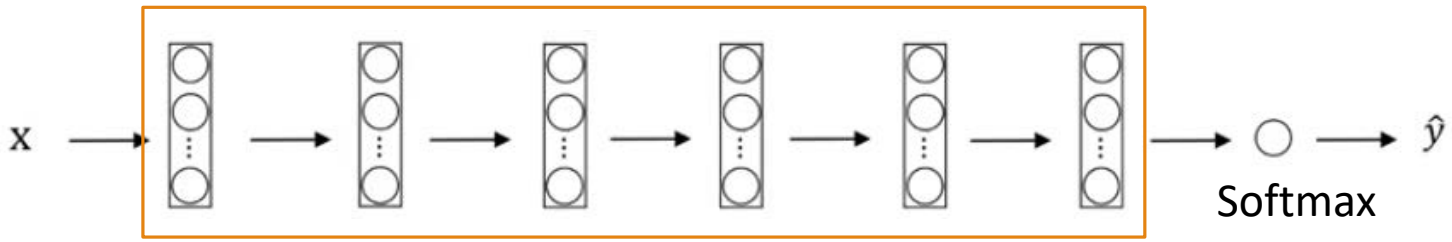
- 121 layers with trainable parameters
- 1 by 1 Conv Layer – Bottleneck layer
  - Deeper architecture
  - Reduce the size of parameters
- Keras Implementation
  - keras.applications.densenet.DenseNet121

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Method (Cont.)

Transfer Learning

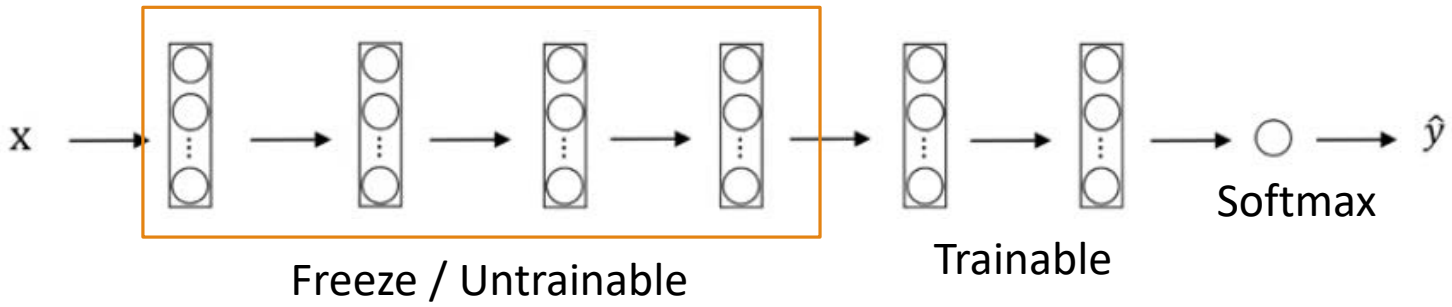
- To improve learning in the target task by leveraging knowledge from the source task
- Popular in computer vision and natural language processing
  - GloVe, Word2vec
  - DenseNet, ResNet, VGG, etc.
- Method
  - Freeze entire model and only train the classification part (Small training set)



Method (Cont.)

Transfer Learning

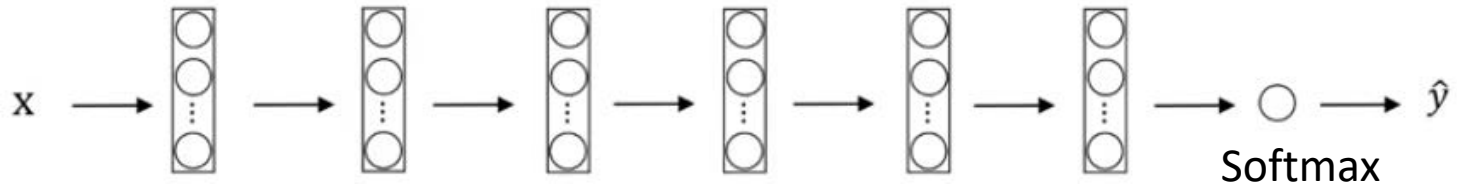
- To improve learning in the target task by leveraging knowledge from the source task
- Popular in computer vision and natural language processing
  - GloVe, Word2vec
  - DenseNet, ResNet, VGG, etc.
- Method
  - Freeze entire model and only train the classification part (Small training set)
  - Freeze most layers of the model and train change the classification part (Medium training set)



Method (Cont.)

Transfer Learning

- To improve learning in the target task by leveraging knowledge from the source task
- Popular in computer vision and natural language processing
  - GloVe, Word2vec
  - DenseNet, ResNet, VGG, etc.
- Method
  - Freeze entire model and only train the classification part (Small training set)
  - Freeze most layers of the model and train change the classification part (Medium training set)
  - Train entire model (Large training set)



Modeling

Data Preprocessing

- Remove invalid images
  - 'dd6dfed324f9fcb6f93f46f32fc800f2ec196be2'
  - '9369c7278ec8bcc6c880d99194de09fc2bd4efbe’
- Data augmentation
  - Keras image preprocessing - ImageDataGenerator class
- Balance training dataset
  - Tumor: 80,000
  - No-Tumor: 80,000

Environment

- Kaggle Kernel with GPU

Modeling – Baseline

Base CNN Model

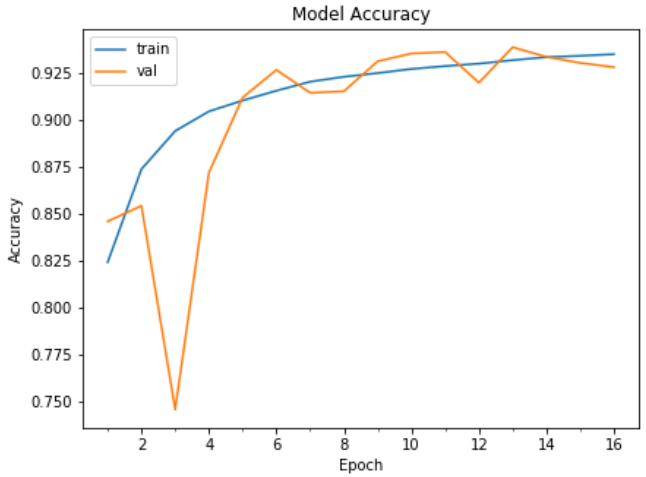
- 14 layers with trainable parameters
- Batch normalization after each Conv2D layer
- Training Setup
  - Training steps: 4,500
  - Validation steps: 500
  - 20 Epochs with early stopping
- 5.01 Million Parameters

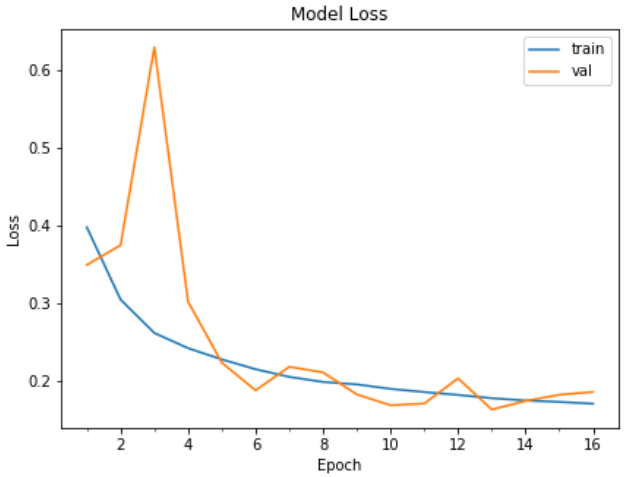
```
Total params: 5,008,129
Trainable params: 5,006,785
Non-trainable params: 1,344
```

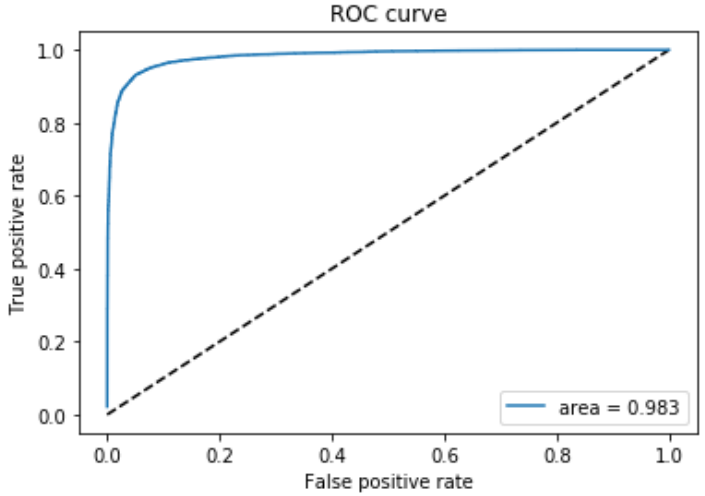
Results - Baseline

Base CNN Model

- Validation Accuracy: 98.30%
- Run Time: 368s per epoch (around 6 minutes)







Modeling – ResNet50

ResNet50

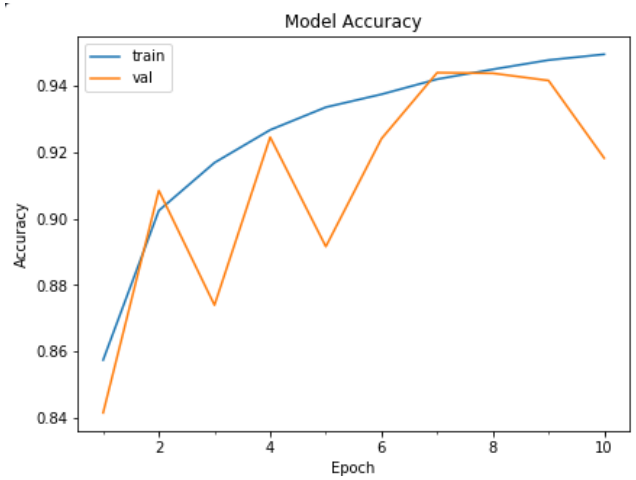
- 50 + 3 layers with trainable parameters
- Batch normalization
- Training Setup
  - Training steps: 4,500
  - Validation steps: 500
  - 10 Epochs with early stopping
- 23.59 Million Parameters

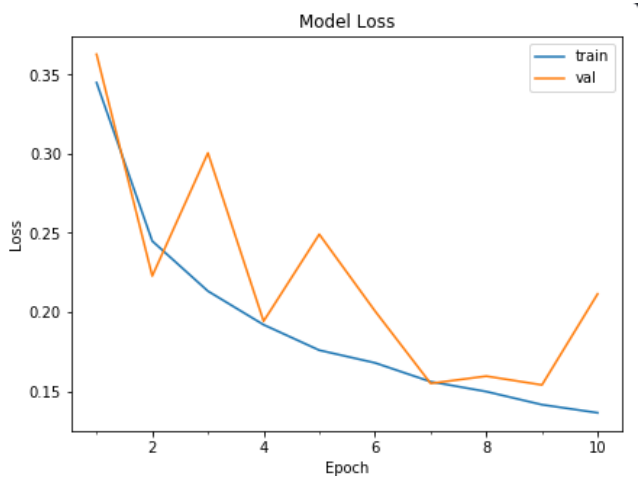
```
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

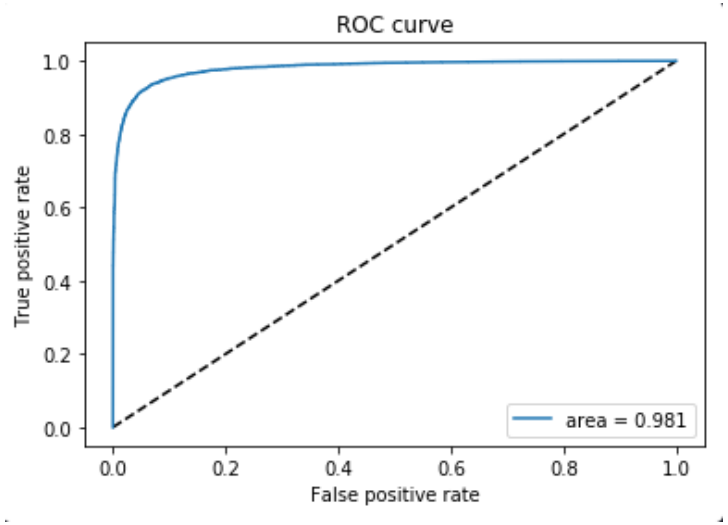
Results – ResNet50

ResNet50

- Validation Accuracy: 98.06%
- Run Time: 1738s per epoch (around 29 minutes)







Modeling – DenseNet121

DenseNet121

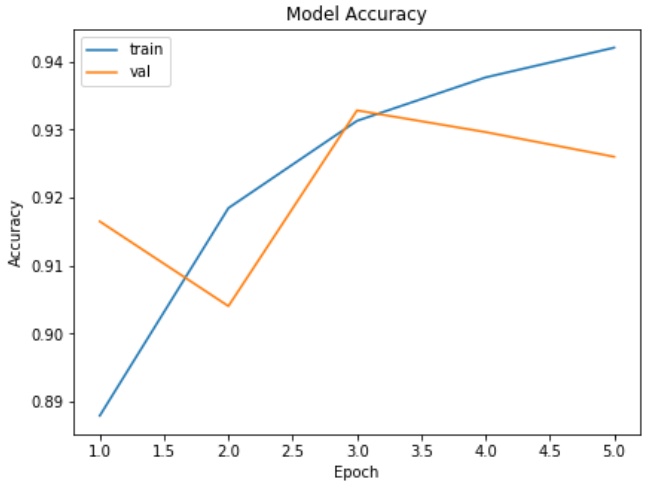
- 121 + 3 layers with trainable parameters
- Batch normalization
- Training Setup
  - Training steps: 4,500
  - Validation steps: 500
  - 10 Epochs with early stopping
- 9.40 Million Parameters

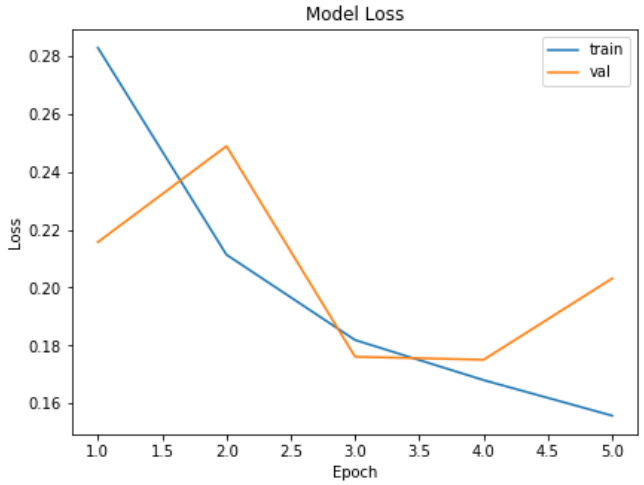
```
Total params: 9,398,081
Trainable params: 9,313,921
Non-trainable params: 84,160
```

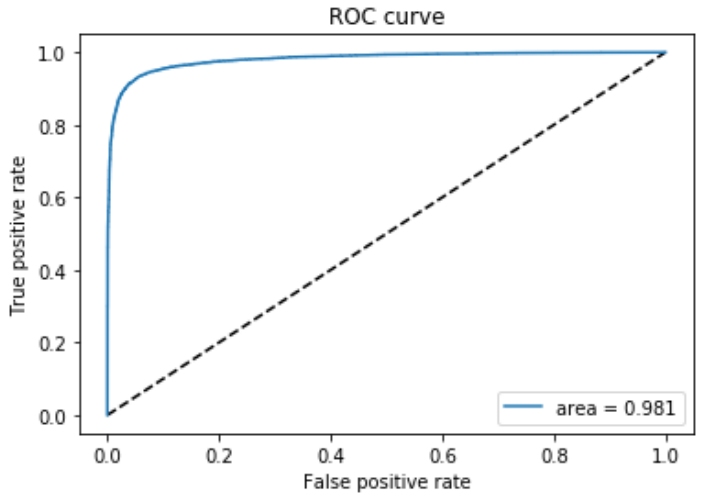
DenseNet121

DenseNet121

- Validation Accuracy: 98.08%
- Run Time: 1512s per epoch (around 25 minutes)







Results - Summary

Model	# of Layers	Run Time (/epoch)	Parameter Size	Validation Accuracy
Base CNN	14	368s	5.01M	98.30%
ResNet50	53	1738s	23.59M	98.06%
DenseNet121	124	1512s	9.40M	98.08%

Conclusion

Application of Image Classification

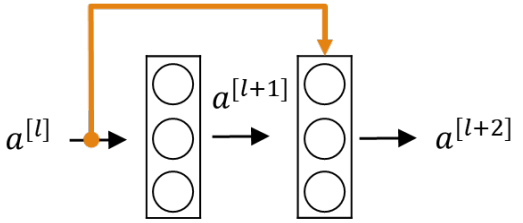
- Cancer detection

Advanced CNN Architecture

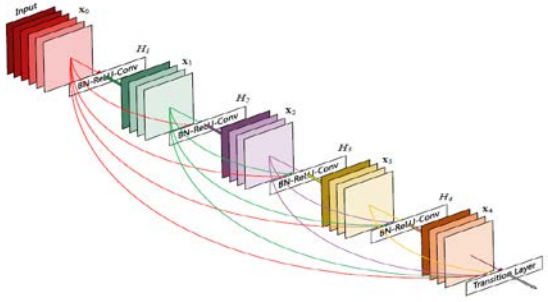
- Residual Neural Network
  - ResNet50
- Densely Connected Network
  - Dense121
- 1 by 1 Convolutional Neural Network
  - Bottleneck layer
  - Reduce the size of parameters
  - Commonly used in complex CNN architecture

Transfer Learning

- “Standing on the shoulders of giants.”



Residual Block



Dense Block

Reference

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017, July). Densely connected convolutional networks. In *CVPR* (Vol. 1, No. 2, p. 3).

Convolutional Neural Networks by deeplearning.ai. https://www.coursera.org/learn/convolutional-neural-networks/home/welcome