Deep Learning - From Theory to Practice

Lawrence O. Hall

Department of Computer Science and Engineering

University of South Florida

Tampa, FL 33620

USA

lohall@mail.usf.edu

Why Deep Learning?

- For some types of data (images, perhaps voice) deep learned models **automatically extract features**
- Deep learned models can work with lots of labeled data very effectively
- Deep learned models have shown **very high accuracy** in many domains

Why Deep Learning?

- For object recognition in scenes, deep neural networks are far and away the best approach
  - **Think autonomous cars**
- Face recognition under good lighting conditions – deep neural nets
- Voice recognition – deep neural nets
- Language translation – deep neural nets

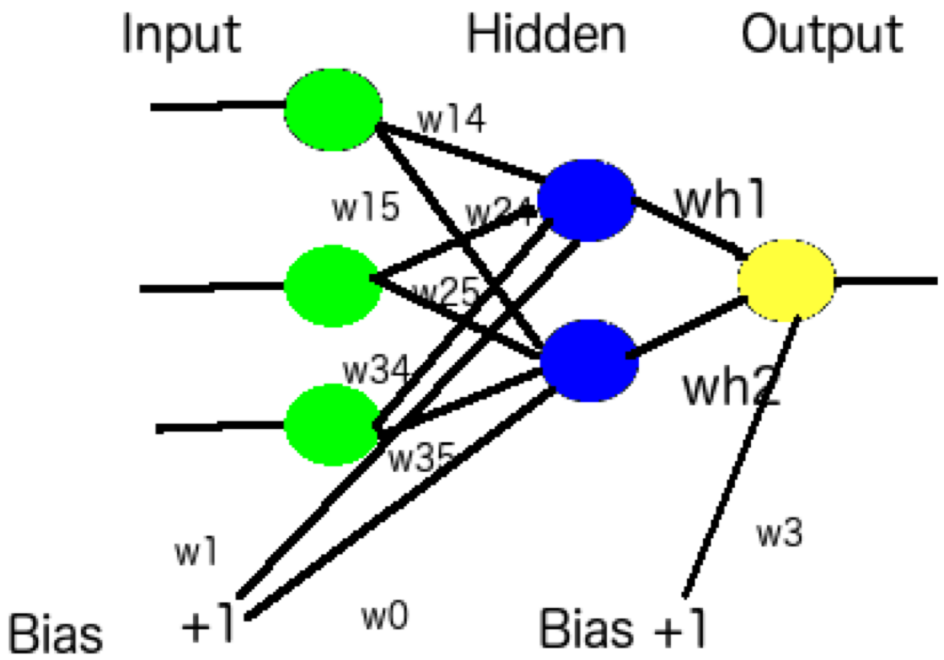
What is Deep Learning?

- Learns a neural network model with more than 1 hidden layer (a layer that is not input or output)
- Most of the "big" results have used lots of **labeled** data to build models that predict classes
- Can use deep learning for regression or (less well studied) grouping unlabeled data, i.e. clustering

Neural network learning

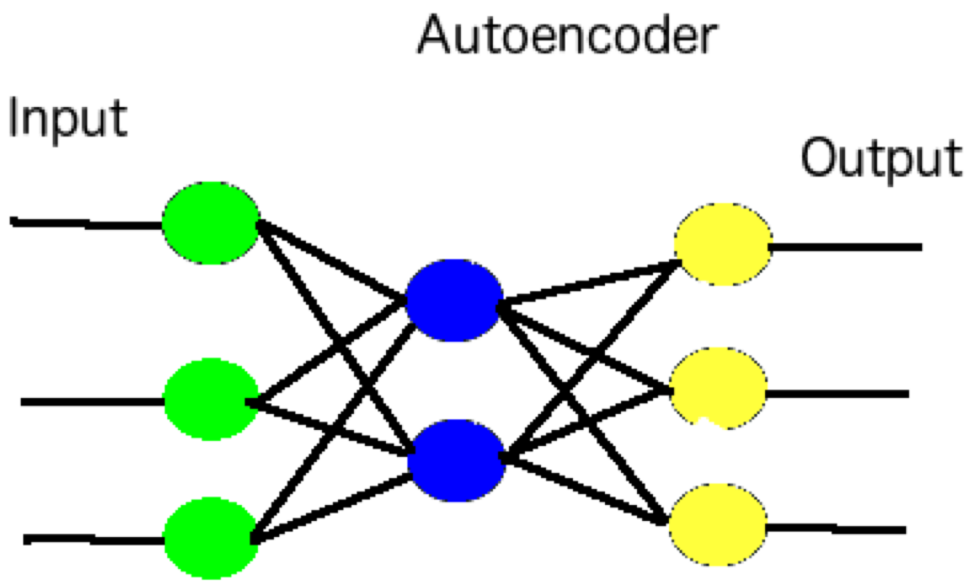
- The focus here is on labeled data
- So, we have inputs (maybe features, maybe images, etc.) and some label(s) for each training example
- There are nodes or units in the network connected in some way (typically called feedforward as the “data flows” from inputs to outputs)
- It is necessary to ”learn” the weights on the connections between nodes

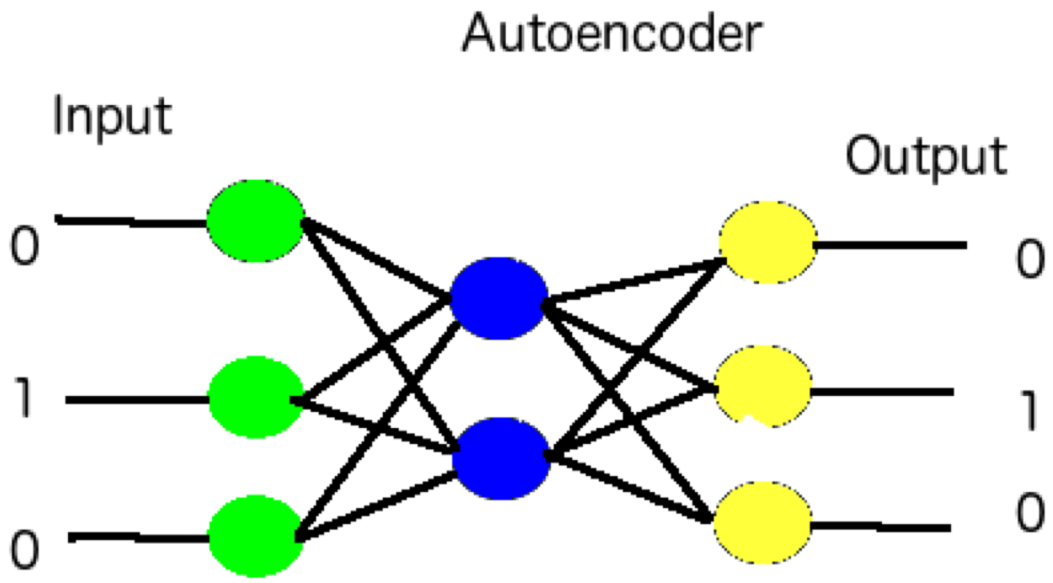
A one hidden layer neural network – Universal Approximator

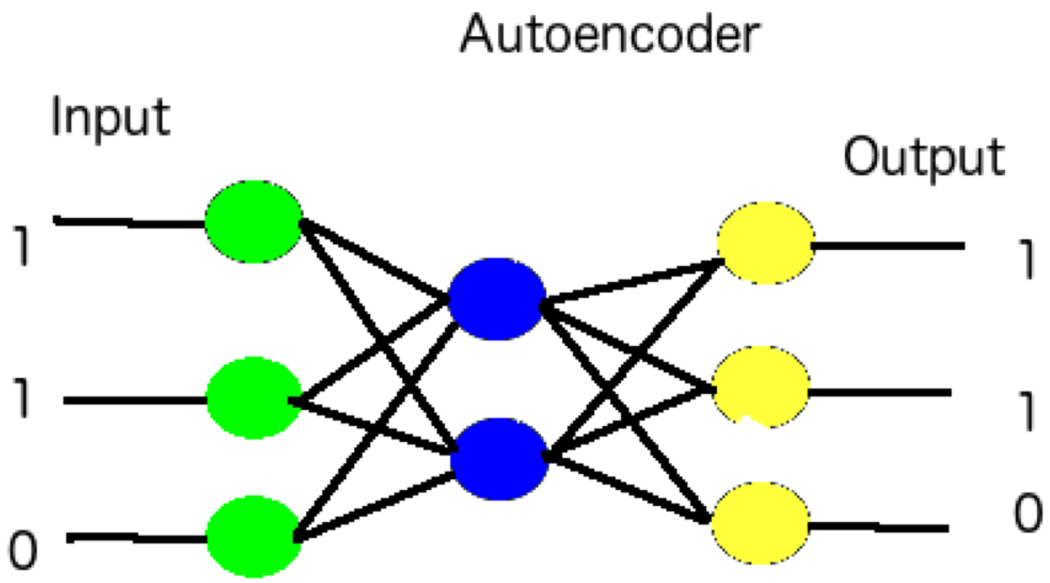


Autoencoders

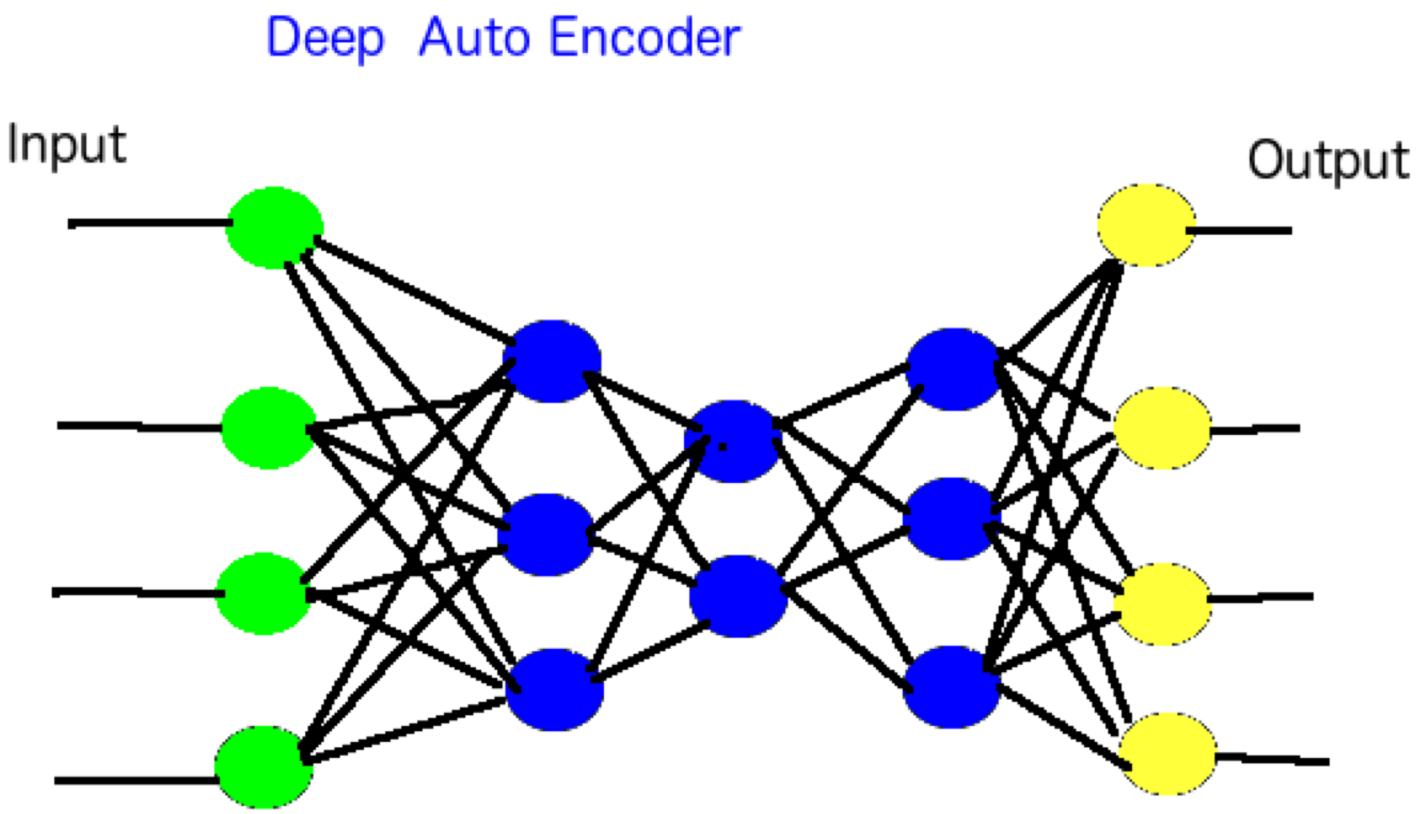
- Neural networks that encode a representation of their inputs in their hidden layer(s)
- The outputs repeat the network inputs
- The number of hidden units is typically smaller than the number of inputs


Autoencoder
Input
Output

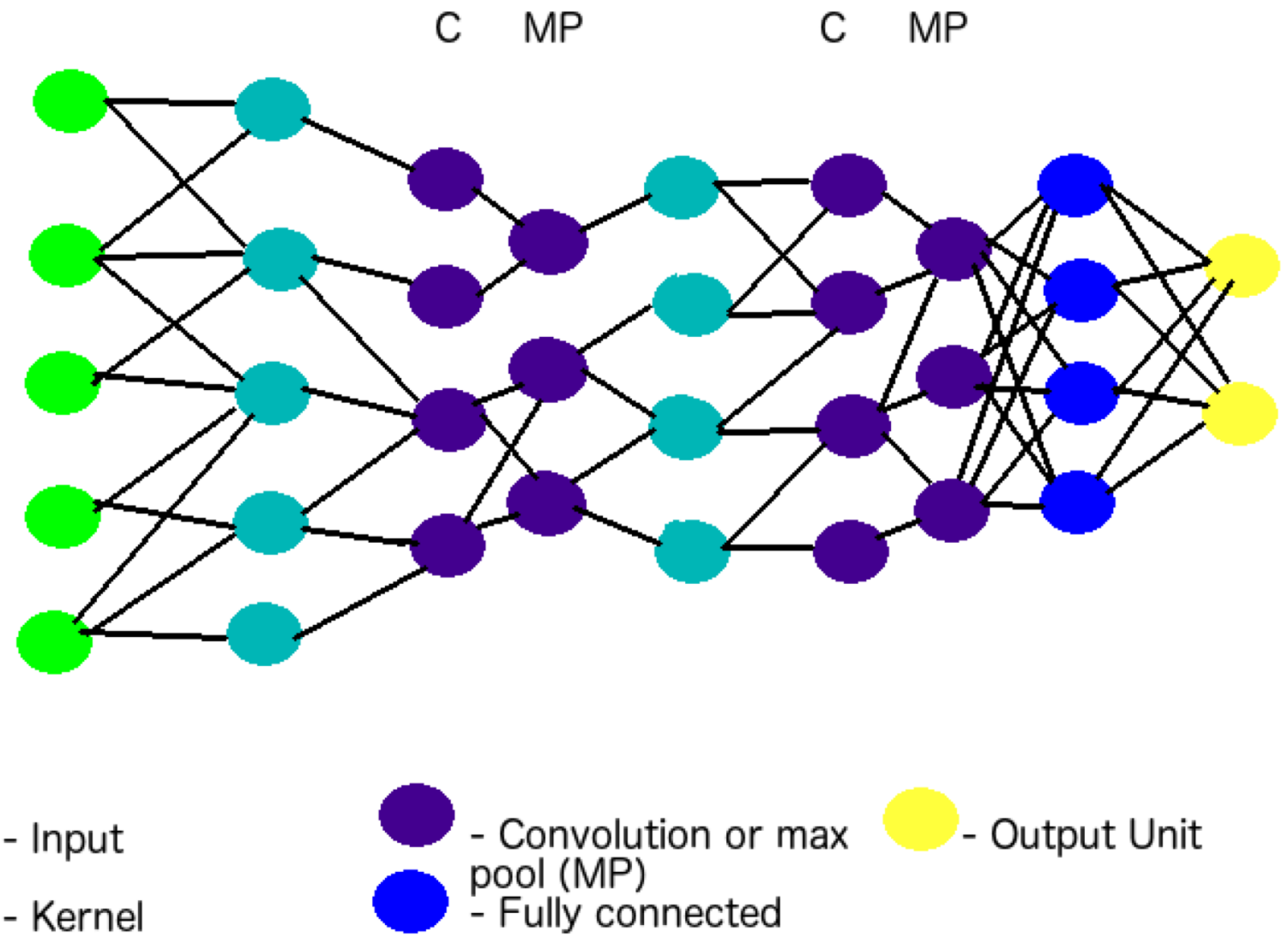

Autoencoder
Input
Output
0
1
0
0
1
0


Autoencoder
Input
Output
1
1
0
1
1
0

More than 1 hidden layer is a **deep neural network**. All links between inputs and outputs have learnable weights on them



A deep convolutional neural network (CNN)



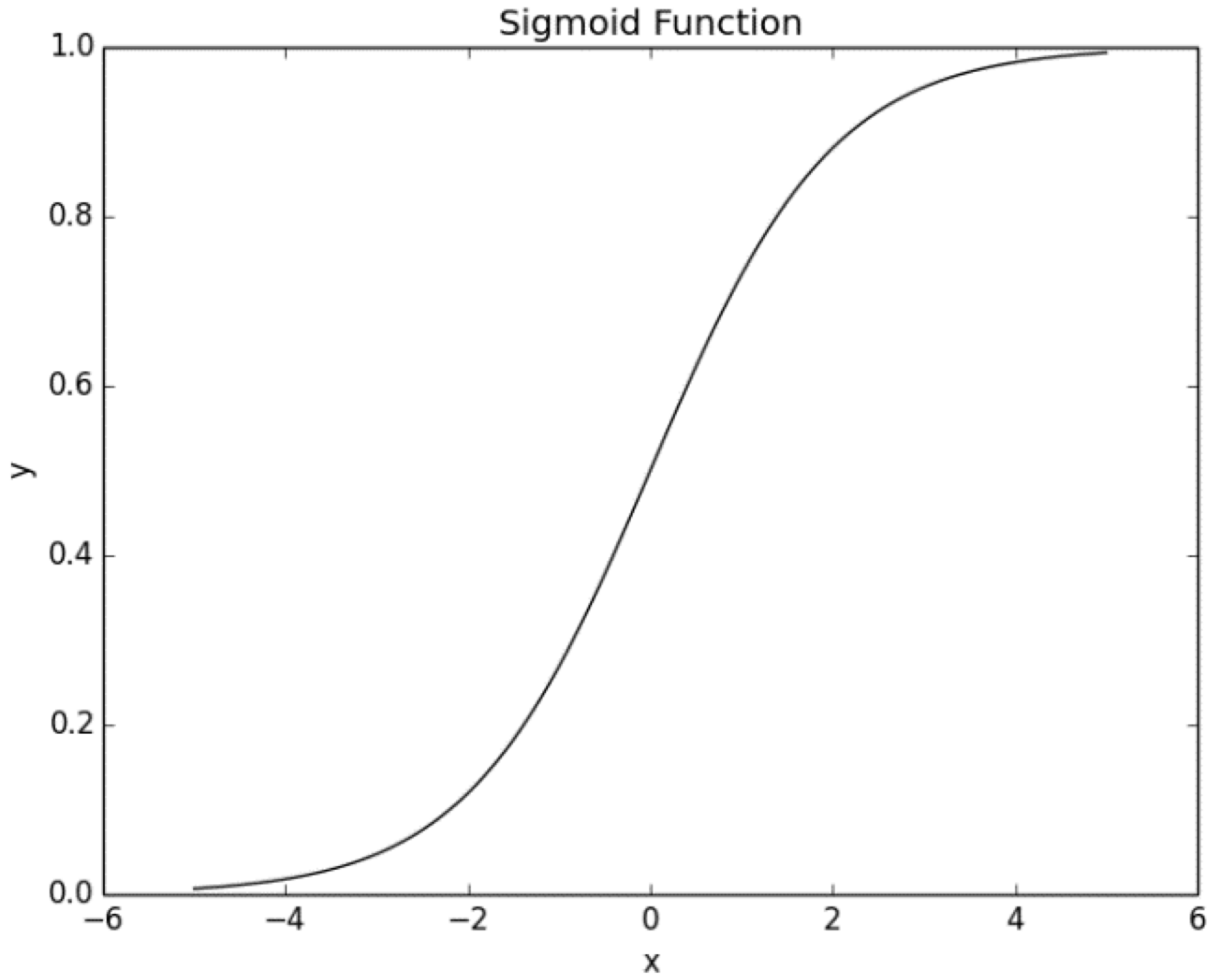
How do we learn in a deep neural network

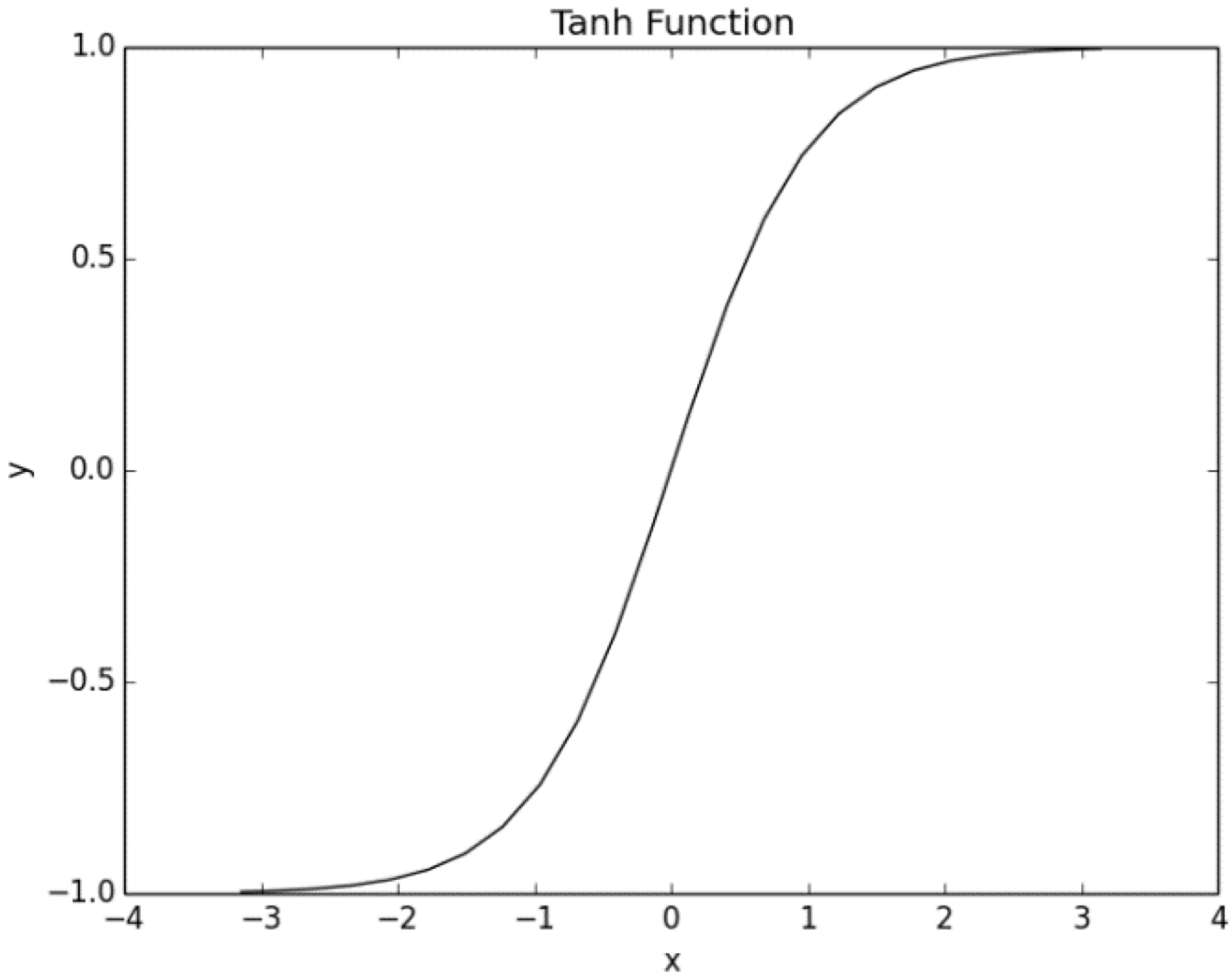
- Variants of backpropagation
- Our problem is we do not know (exactly) what the correct outputs of hidden units are
- However, backpropagation allows us to learn all weights
- A basic primer follows

We need a non-linear activation function

- Each node or unit will sum its weights times the activations at the previous layer and then apply a function to this
- Let weights be w_i and activations be a_i so for a unit/node we have
- net= $\sum_i a_i w_i$, with an activation function f(net)

- We need a differentiable function that is non-linear. Many options: **sigmoid**, hyperbolic tanget, etc.


Sigmoid Function
1.0
0.8
0.6
0.4
0.2
0.0
y
−6
−4
−2
0
2
4
6
x


Tanh Function
1.0
0.5
0.0
−0.5
−1.0
y
−4
−3
−2
−1
0
1
2
3
4
x

Our derivation will use the sigmoid activation function

- The input to a node/unit: $net = x = \sum_i a_i w_i$
- Sigmoid: $f(x) = 1/(1 + e^{-x})$
- $\frac{df(x)}{dx} = f(x)(1 - f(x))$

Error gradient for a Sigmoid Unit

- Let t_k be true output, o_k be observed unit output and $E = \frac{1}{2}\sum_{k \in D}(t_k - o_k)^2$ be the loss or error.
- $\partial E / w_i = \frac{\partial}{w_i}\frac{1}{2}\sum_{k \in D}(t_k - o_k)^2$
- $= \frac{1}{2}\sum_k 2(t_k - o_k)\frac{\partial}{w_i}(t_k - o_k)$
- $= \sum_k (t_k - o_k)(-\frac{\partial o_k}{w_i})$
- $= -\sum_k (t_k - o_k)(\frac{\partial o_k}{net_k}\frac{\partial net_k}{w_i})$

Error gradient for a Sigmoid Unit, continued

- $\frac{\partial o_k}{net_k} = \frac{\partial f(net_k)}{\partial net_k} = f_k(1 - f_k)$
- $\frac{\partial net_k}{w_i} = \frac{\partial(\vec{w} \cdot \vec{x_k})}{\partial w_i} = x_{i,k}$
- $\frac{\partial E}{w_i} = - \sum_{k \in D} (t_k - o_k) f_k (1 - f_k) x_{i,k}$

Backpropagation Algorithm

Initialize all weights to small random numbers.
Repeat until error is *low enough*

1. Submit input training example x_k to network and get network outputs
2. For each output unit m
$$\delta_m \leftarrow o_m(1 - o_m)(t_m - o_m)$$
3. For each hidden unit h
$$\delta_h \leftarrow f_h(1 - f_h) \sum_{m \in outputs} w_{h,j} \delta_m$$
4. Update each weight $w_{i,j}$ as $w_{i,j} + \Delta w_{i,j}$
where $\Delta w_{i,j} = \eta \delta_j x_{i,j}$

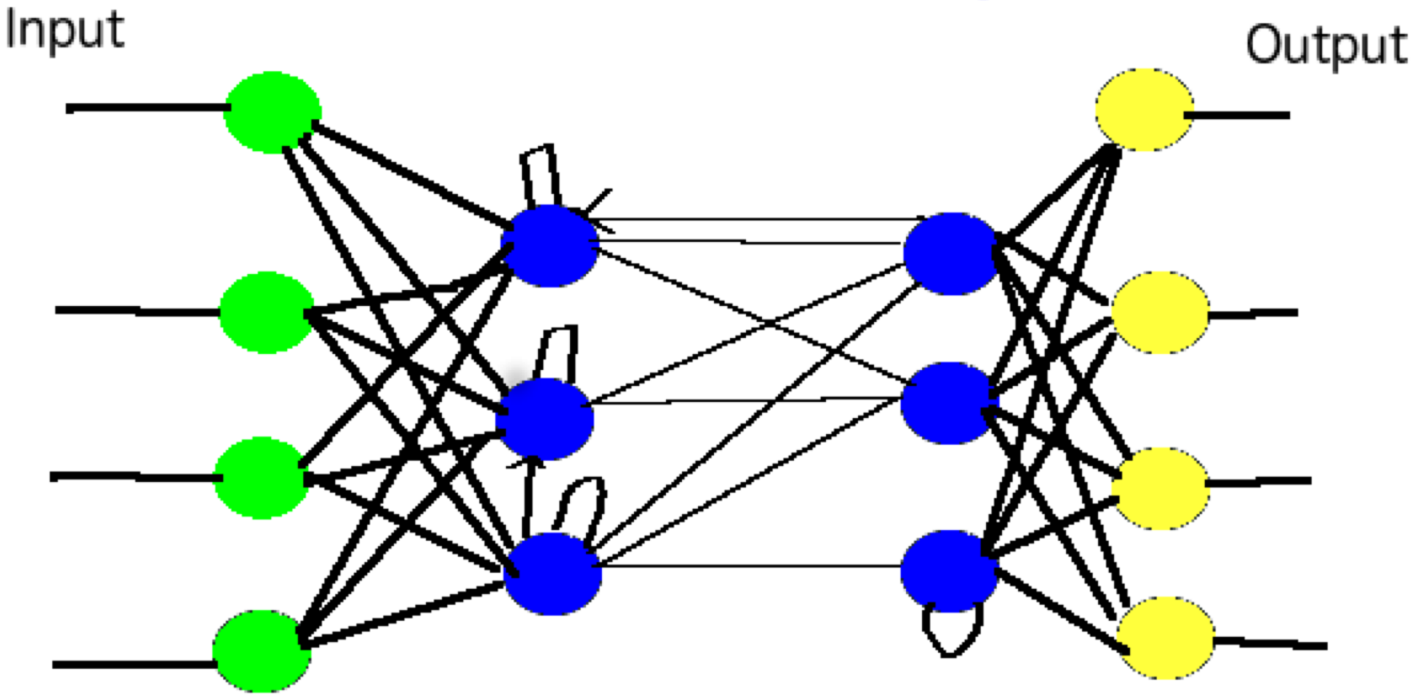
Backpropagation continued

- Our learning rate η should generally be small.
- Often there is a momentum term α using the weight change at time $z-1$

$$\Delta w_{i,j}(z) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(z-1)$$

- We can accumulate weight changes over a batch of (to all) examples
- With more than 1 hidden layer, just recursively apply backpropation. So, it is all you need to learn, though there are **MANY** variations.

Recurrent Deep Neural Network with self loops
and backward links between layers



Recurrent Neural Networks

- Essentially have an internal state
- Great for speech recognition or machine translation where there are sequences of data
- Our example will be long short term memory (LSTM) recurrent networks which have been successful in unconstrained handwriting recognition as well

Recurrent Neural Networks

- Can calculate loss (error) at each time step or after a sequence
- With the same matrix at each time step our gradient for weight update can go to 0 (vanish) or get really large (explode)
- Can keep the weights small through L_1 or L_2 regularization
- If gradient gets too big, cut it down called clipping

LSTM Networks

- Let **W** be a weight matrix, b be bias and **x** be an input vector, o be an output
- LSTM networks apply linear matrix operations to current input, hidden units from the previous time step and use the resultant linear terms as arguments of activation functions f
- $h_t = f(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h h_{t-1} + b_h)$
- $o_t = f(\boldsymbol{W}_o h_t + b_o)$

- The same matrix $\boldsymbol{U}_h$ each time step

LSTM Networks

- Designed to address vanishing gradient problem
- It has ”memory cells” which are a combination of hidden units and elementwise products and sums between units that implement *gates* to them
- Memory cells can retain information for long periods of time
- Their input and output gates are a function of current input and hidden units at previous time step

LSTM Networks

- The original LSTM had input and output gates. Later forget gates and peephole weights were added
- We look at the most popular variant with all gates
- At time step t input $\boldsymbol{i}_t$, forget $\boldsymbol{f}_t$, and output $\boldsymbol{o}\boldsymbol{g}_t$ gates receive current inputs $\boldsymbol{x}_t$, and previous hidden unit information $\boldsymbol{h}_{t-1}$

LSTM Networks

- At each time step t, the input gates determine whether a potential input

$s_t = tanh(\boldsymbol{W}_s \boldsymbol{x}_t + \boldsymbol{U}_s h_{t-1} + b_s)$ is important enough to go in memory unit c_t

- Forget gates $\boldsymbol{f}_t$ allow the contents of memory units to be erased.

- Output gates og_t determine whether $\boldsymbol{y}_t$ the content of memory units transformed by activation functions should go into hidden units $\boldsymbol{h}_t$

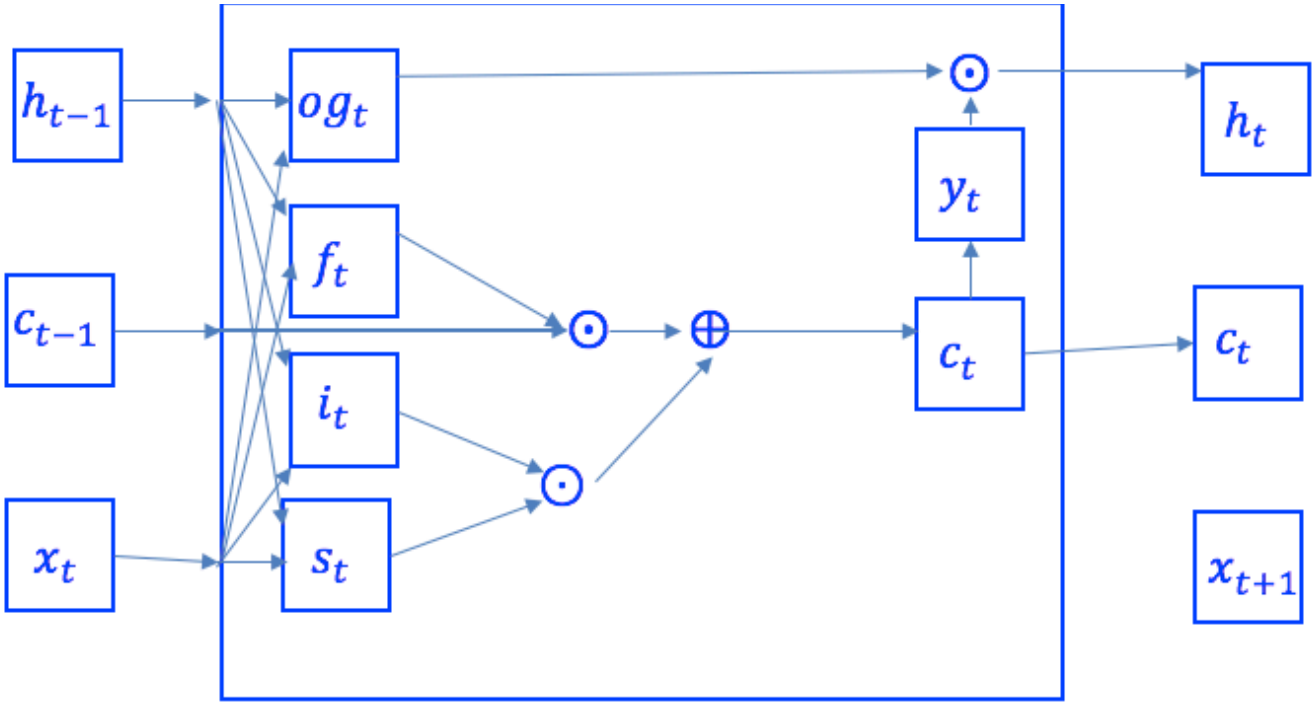
LSTM Networks – final gating

- $\boldsymbol{h}_t = og_t \circ y_t$ is the elementwise product between the output gate and transformed memory contents, where $y_t = \tanh(c_t)$

- Memory units are updated as:

$$c_t = f_t \circ c_{t-1} + i_t \circ s_t$$

Components of LSTM	RNN
LSTM Unit Output	$h_t = og_t \circ y_t$
Output Gate units	$og_t = \sigma(\mathbf{W}_{og}x_t + \mathbf{U}_{og}h_{t-1} + b_{og})$
Transformed mem. cell contents	$y_t = tanh(c_t)$
Gated update to mem. cell units	$c_t = f_t \circ c_{t-1} + i_t \circ s_t$
Forget gate units	$f_t = \sigma(\mathbf{W}_f x_t + \mathbf{U}_f h_{t-1} + b_f)$
Input gate units	$i_t = \sigma(\mathbf{W}_i x_t + \mathbf{U}_i h_{t-1} + b_i)$
Potential input to memory cell	$s_t = tanh(\mathbf{W}_s x_t + \mathbf{U}_s h_{t-1} + b_s)$



Structure of LSTM Unit

Deep Learning Strengths

- Works really well with big data (with labels)
- Big successes in image recognition
  - ImageNet – 14,000,000+ images in over 1000 categories
    - Deep learning has been significantly more accurate than any other approach for ImageNet
- Very good at document recognition (lots of documents available)
- Good at translation (Google Translate has improved a lot, error rate down by 60%)
  - Can create big corpus for translation

Images from ImageNet

Some of the 1000 classes of images

Deep Learning Strengths

- Deep convolutional neural networks do feature selection for you!
  - No extraction and selection of features -> less up front work
- Feature selection can also be done in deep recurrent neural networks
- May want convolutional layers for images

Images as input – Simple Example

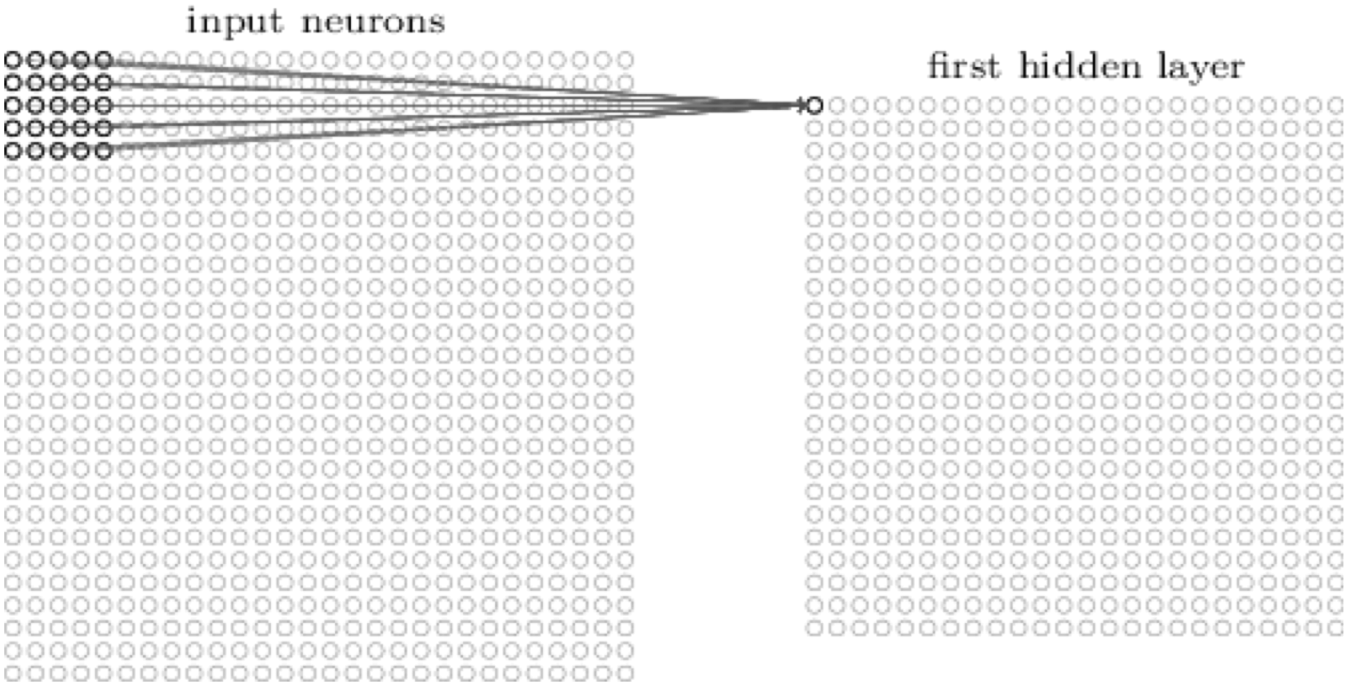
- Think of 28x28 images of handwritten characters (MNIST):

- Let's just use the pixel intensities as features (784)

- A training set of 60,000 examples and separate test set of 10,000 examples exist.

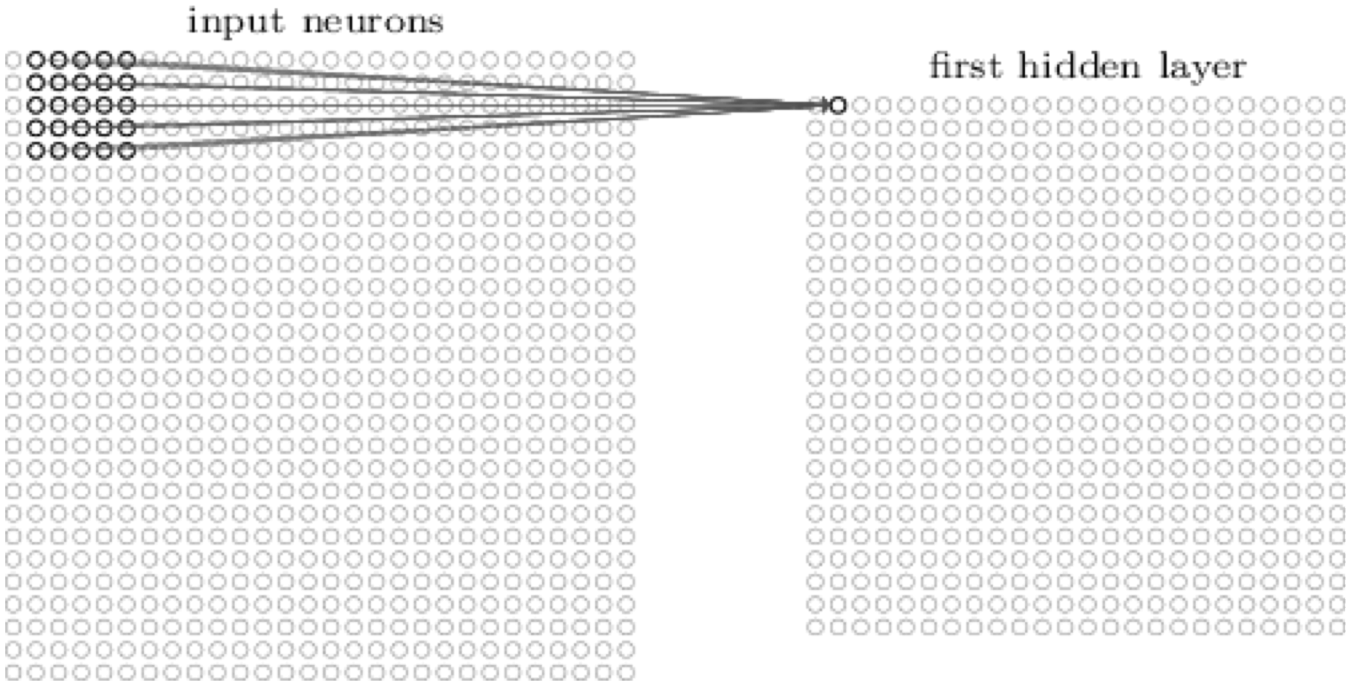
Using convolution

- Let's use 5x5 "local receptive fields" and slide them by a stride of 1.
- Gives us a 24x24 convolutional hidden layer.



Using convolution

- Let's use 5x5 "local receptive fields" and slide them by a stride of 1.
- Gives us a 24x24 convolutional hidden layer.



Some details

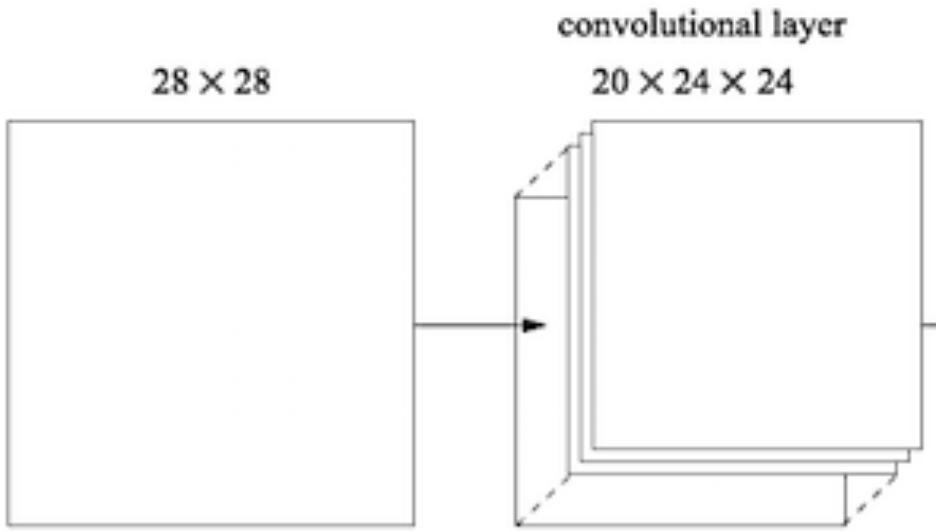
- Each hidden unit in the convolutional layer has a bias and 25 weights attached to it.
- The weights and bias will be the same for all 576 hidden units. Formally, the output of the j,kth hidden neuron for activation function f is:

$$f\left(b+\sum_{l=0}^{4}\sum_{m=0}^{4}w_{l,m}o_{j+l,k+m}\right)$$

- Where b is the bias, **w** is the vector of 25 weights and **o** is the output of the appropriate neuron from the window in the convolutional layer.

Convolutional layer

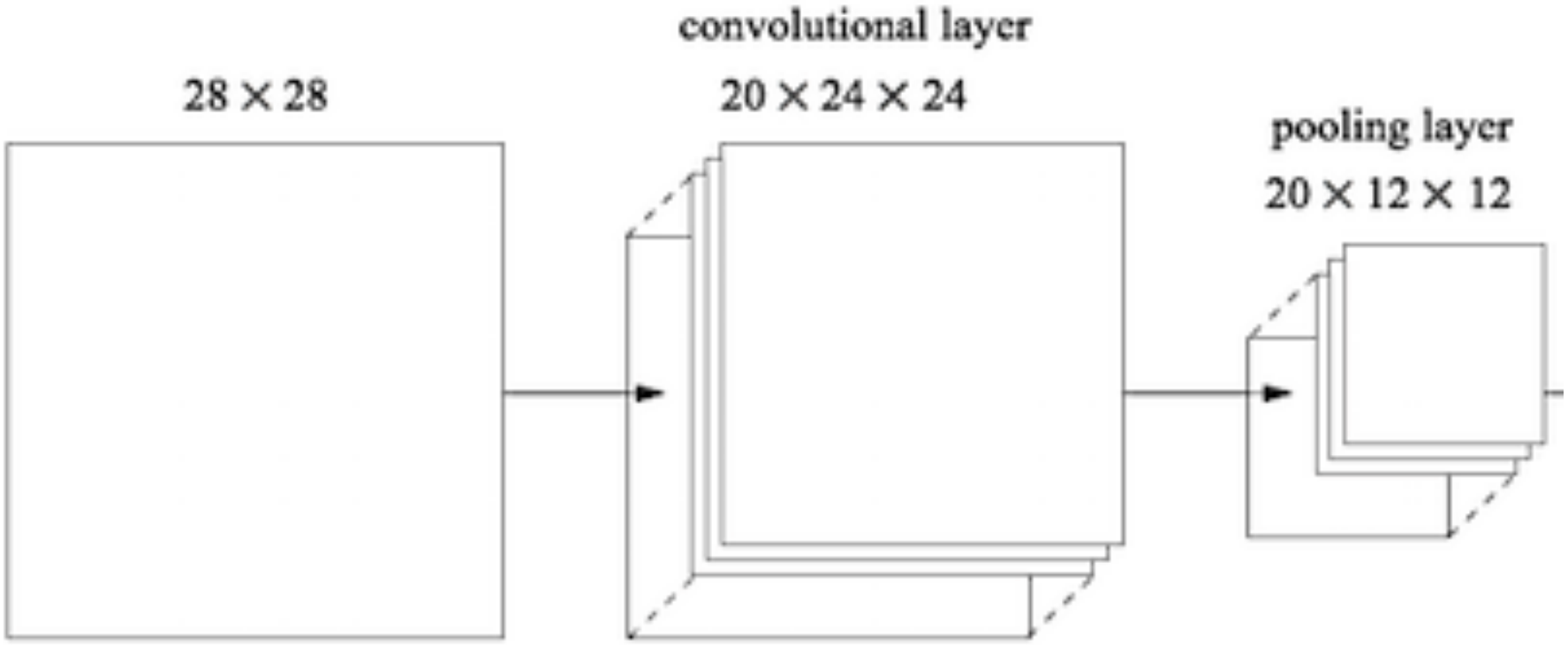
- We can think of a single convolution as pulling out one feature from an image
- So, maybe in the hidden layer we need n convolutions. Below n= 20.



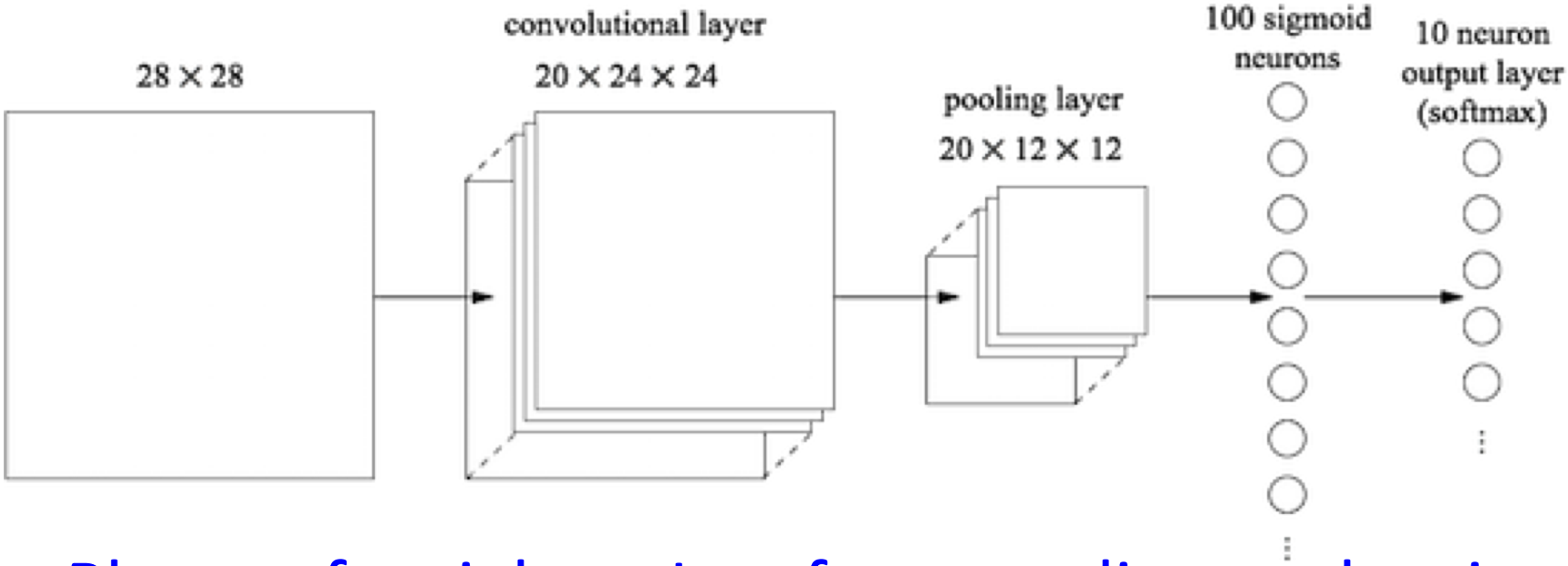
Max Pooling layer

- Pooling layers are used after convolutional layers to simplify or reduce the information
- They will be applied to each of the convolutions so we would have 20 applications in this example
- Often a 2x2 window is used (so 24x24 gets reduced to 12x12). Non-overlapping (or stride of 2)
- The max part is that the maximum value from the window is all that is passed on

Max Pooling layer

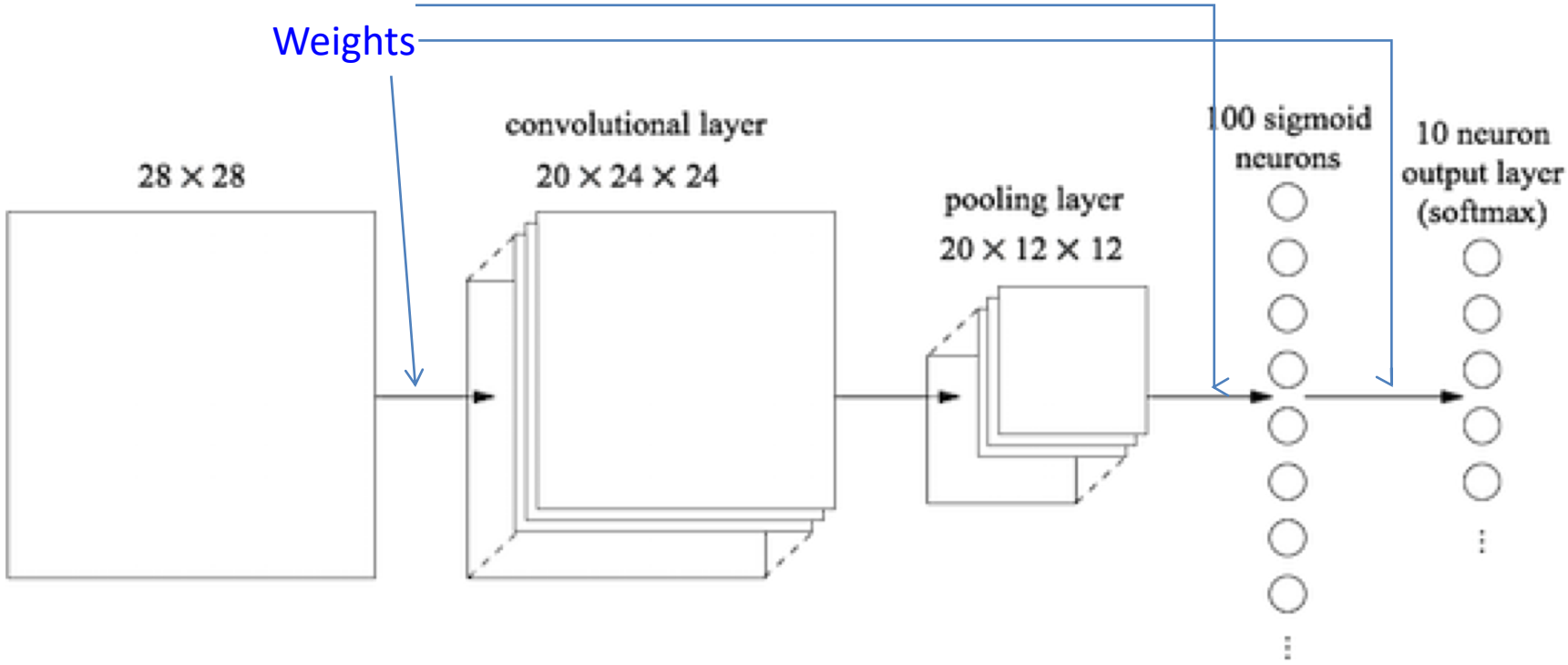


Small 5-layer CNN for MNIST images



Plenty of weights. Just from pooling to the sigmoid layer there are 2880 weights to each neuron (so 288,000 weights). How many overall to learn?

Small 5-layer CNN for MNIST images



How many overall to learn?

520 + 288,000 + 1,000 = 289,520

MNIST Results

- Elastic distortions allow training data augmentation by creating realistic, possible images
- Large/deep convolutional neural net (elastic distortions) 0.35 error on test data (vs. 0.95 for no distortions and an early CNN
- Ensemble of 35 convolutional networks (elastic distortions) 0.23 error

Imagenet Results and Architectures

- **Large Scale Visual Recognition Challenge – Done yearly with 1000 categories of RGB images**
- **Training data is: 10,000,000 labeled images depicting 10,000+ object categories**
- **Test data for 2017 was: 150,000 photographs, collected from flickr and other search engines, hand labeled with the presence or absence of 1000 objects**
- **Validation data was a random 50,000 from the 150,000 examples**

Large Scale Visual Recognition Challenge

Name	Layers	Top-5 Error %
AlexNet (2012)	8	15.3
VGG Net (2014)	19	7.3
ResNet (2016)	152	3.6
Squeeze and Excitation ResNet – Ensemble (2017)	152	2.25

VGG-B-net Architecture

- 3x3 kernel or receptive field for all convolution layers, stride of 1 and padding of 1, so no reduction in image size
- Max pooling with a 2x2 window with a stride of 2
- Two fully connected layers of 4096 units with softmax outputs
- Rectified linear unit (RELU) activation function. All negative sums made 0

VGG-B-net Architecture

Layer	
Input	224x224x3
Convolutional	3x3x64 3x3x64
maxpool	2x2
convolutional	3x3x128 3x3x128
maxpool	2x2
convolutional	3x3x256 3x3x256
maxpool	2x2
convolutional	3x3x512 3x3x512
maxpool	2x2

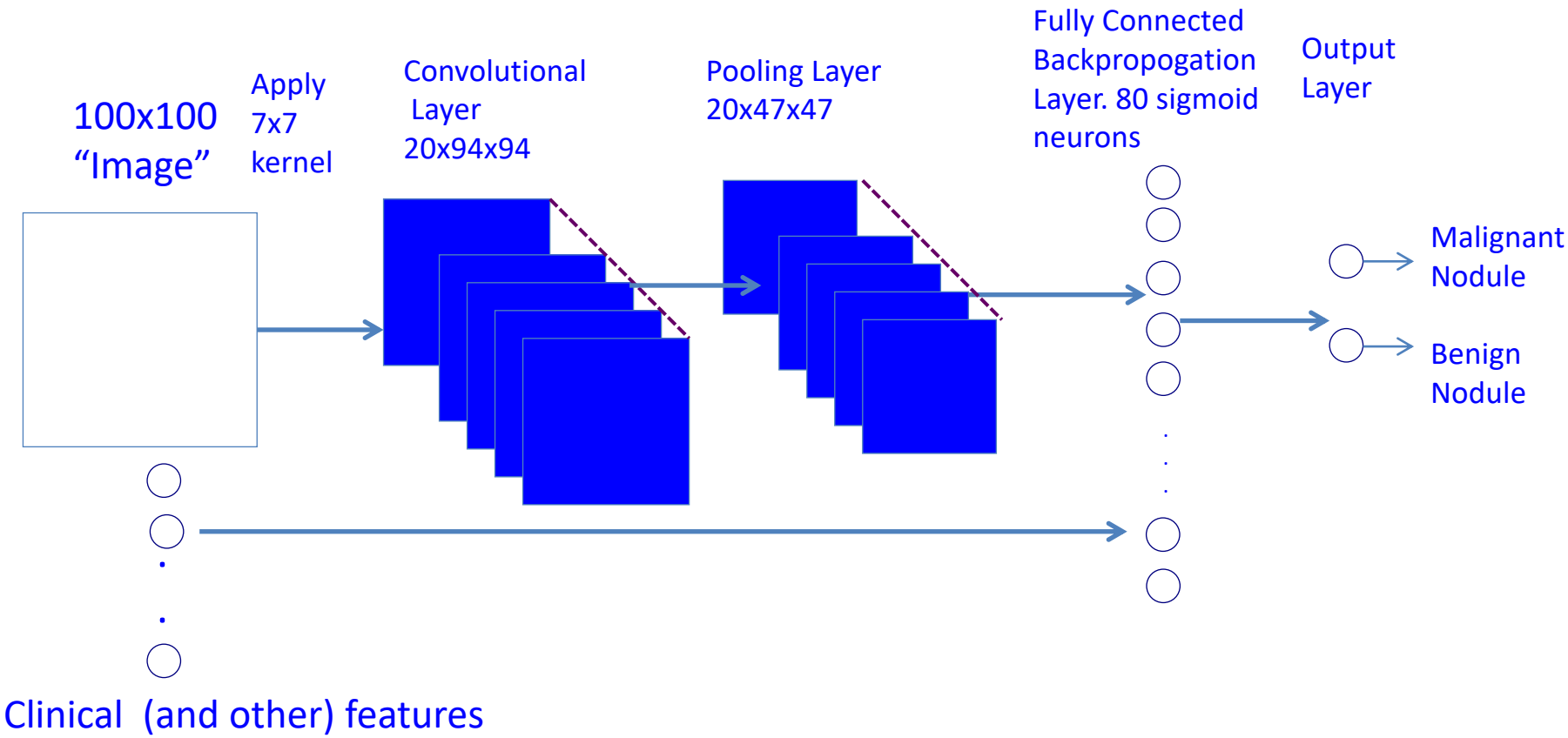
VGG-B-net Architecture - Continued

Layer	
Fully connected	4096
Fully connected	4096
Output - Softmax	1000

ResNet-50 Architecture (224x224x3 input). Top 5 Error 5.25%

	7x7x64 stride 2 **3x3 maxpool stride 2**
	1×1, 64 3×3, 64 1×1, 256 } x 3
	1×1, 128 3×3, 128 1×1, 512 } x 4
	1×1, 256 3×3, 256 1×1, 1024 } x 6
	1×1, 512 3×3, 512 1×1, 2048 } x 3
	average pool, 1000-d fully connected, softmax

Small Medical Image Model



Quick Summary

- CNN's take a raw image as input. Then convolution and max pooling are typically applied
- You can have multiple convolutions followed by max pooling
- You can have multiple "traditional" hidden units
- You may use dropout in training
  - In dropout, randomly delete nodes/units and their connections during training
  - It is a regularization approach to help with overfitting

Quick Summary

- You WILL have lots of weights to train, 289,520 in our simple MNIST example
- You need lots of examples to train these (so big data), but training will be slow – graphical processing units (GPUs) help
- You can try tuning a deep neural network built for a different, but related problem
- Accuracy is the best for images, text, voice; today. Why?

Quick Summary

- You WILL have lots of weights to train, 289,520 in our simple MNIST example
- You need lots of examples to train these (so big data), but training will be slow – GPU's help
- You can try tuning a deep neural network built for a different, but related problem
- Accuracy is the best for images, text, voice; today. Why?
  - Easier to get lots of labeled data

Options in building a DNN

- The network architecture
  - Convolutional layers
  - How many layers and what type
  - Recurrent layers and what format
- Type of transfer function in the hidden unit: sigmoid, hyperbolic tangent, ReLU, etc.
- Regularization applied
  - Keep weights in check
  - Improve generalization on unseen test data

Options in building a DNN

- Regularization applied
  - Dropout (occasionally randomly remove a node/unit and its connections from being updated)
    - Need a rate
  - Clipping – Don't let a weight get too big
  - Layer normailization
  - Batch normalization
  - Train with mini-batches
  - Weight decay
  - Error functions can regularize
  - L1, L2
  - And many, many more…

Options in building a DNN

- Initialization of weights
  - Random, small weights
- Learning rate (small)
  - Adaptive?
- Momentum term
- Learning variant of backpropagation or something else

Options in building a DNN

- Stopping criterion
  - Use of validation set – preferably
  - Training error or number of epochs (times through data), if not enough data for validation

Keras (https://keras.io/)

- Keras is a high-level neural networks API, written in Python
- Runs on top of top of TensorFlow, CNTK, or Theano
- Designed for fast experimentation

Keras

- Keras allows for easy and fast prototyping (modular, and extensible)
- Supports both convolutional networks and recurrent networks, as well as combinations of the two
- Runs seamlessly on CPUs and GPUs

Keras

- If you want to create a stacked layer model (deep) and have tensorflow, for example, as the backend.
- Below 100 inputs, 64 hidden, 32 hidden and 10 outputs

- from keras.models import Sequential

```
 model = Sequential()
from keras.layers import Dense
 model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=32, activation='sigmoid'))
 model.add(Dense(units=10, activation='softmax'))
```

Keras

- To compile your model:

```
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.SGD(lr=0.001, momentum=0.8))
```

Train

```
# x_train and y_train are Numpy arrays --just like in the
Scikit-Learn API.
 model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Test

```
error = model.evaluate(x_test, y_test, batch_size=128)
```

Deep Learning for Medical Image Understanding

- A focus on deep convolutional neural networks

- Two reasons:
  - 1. They do a form of feature extraction for you and
  - 2. there are existing, available pre-trained networks from large image sets that can be useful to us

Medical Image Understanding

- Goal is to learn to make make both diagnostic and prognostic predictions
- Diagnosis is always going to involve the treating physicians
- Prognosis is going to be in conjunction with treating physicians, but could exceed their abilities

Medical Image Understanding

- Consider looking at an image to predict survival time for Cancer patients
  - Has real importance to a patient
  - Is not done
  - Clinicians use images and clinical data to give survival time predictions when asked
- Predictions of survival time by Physicians are just very slightly better than random guessing!
- An opportunity for Deep Learning!

Radiomics

- Extract quantitative features from clinical medical images to create tools that can aid M.D.'s decision making

- Consider predicting smokers probability of getting Lung Cancer
  - Current methods look at size of lung nodules
  - Radiomics models have shown significant improvement over size, with less false positives, in predicting a nodule will become cancer
  - Important for monitoring and early treatment is lifesaving in lung cancer

Images of Cancer

- The primary examples here will be images of cancers. Work done in conjunction with the H. Lee Moffitt Cancer Center on the USF campus

- An interesting problem: Breast Cancer
  - There are, very recently, multiple neoadjuvant chemotherapy options
    - Can recommendations for the best one be made from images augmented by clinical information like estrogen receptor status?
    - Can complete responders to chemotherapy be recognized from magnetic resonance images?

Medical Images unique features for Deep Learning

- For a given disease, a research group will generally have a small subset of available images due to privacy and other concerns
  - Small data sets are problematic for (deep) learning

- Clinical data is typically collected in non-uniform ways
  - In the US every patient having a computed tomography (CT) scan or MRI scan will have a potentially different field of view
  - Slice thicknesses will vary, reconstructions can be different, segmentations of abnormal regions will be different

Medical Images unique features

- We are interested in identifying and working with the abnormal cases
  - Abnormal cases are reasonably rare
- Consider the US National Lung Cancer Screening Trial (NLST) had about 27,000 cases in the CT arm
  - All heavy current or former smokers, almost all had lung nodules
  - There were only about 250 cases of cancer
  - How many would like a 99% correct classifier?
    - Just guess No Cancer and you have it

Medical Images Unique Features

- It is difficult to collect lots of data
  - Privacy concerns
- The data sets collected are typically skewed towards normals (imbalanced)
- Some image normalization may be necessary and of course the cancers ***will all be different sizes***

How is learning done from Medical Images without Deep Learning?

- Features are extracted from regions of interest in the image
  - A region of interest will be an abnormality (cancerous tumor for our purposes)

- Features may be
  - texture (Laws, Haralick, LBP, GLRLM, etc.)
  - Shape (volume, spiculations, concavity, etc.)
  - Intensity (histograms of intensities, average intensity, standard deviation, etc.)

How is learning done from Medical Images without Deep Learning?

- You can extract way over 1000 features from an image
  - Which are the useful ones for your problem?
    - Feature selection needed
    - Correlated features?

- After feature extraction and selection, the feature vectors and class labels are given to a classifier to learn a predictive model

Classifiers

- Random Forests
  - An Ensemble of Decision trees where the best feature for each test is chosen from a random set of features
- Support Vector Machines – with kernels can create nonlinear boundaries using just Support Vectors
- Logistic Regression
  - Popular in medicine because you can get an idea of the importance of features
- Others – Decision Trees, Naïve Bayes, etc.

Medical Images and Deep Learning

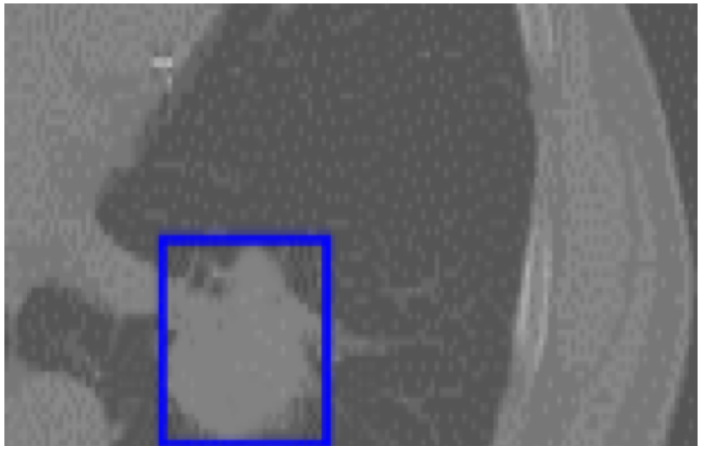
- High accuracy is great!
- Need lots of data to train because there are lots of weights
  - Uh oh, we lack lots of data
- Are there issues with unbalanced classes?
  - Likely (not well studied), but medical cases are unbalanced
- Generally, uniform image sizes are best (or a few different sizes)
  - In Cancer tumors can be BIG, Small or In Between, but few are the same size

Medical Images and Deep Learning

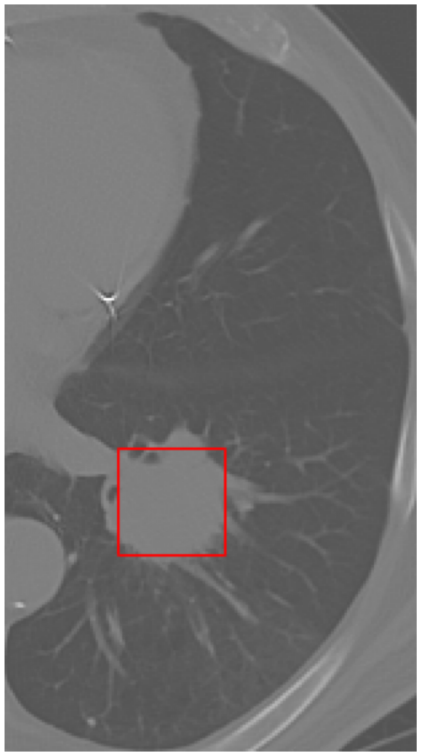
- Cancerous tumors are 3-D and many extracted features are 3-D

- The easiest way to use deep learning is on a 2D slice of the tumor
  - Should be the largest slice
  - However, what if the shape changes are in the direction not captured?
  - Can create a 3D deep neural network with more weights

Exact and Cropped Lung Tumors



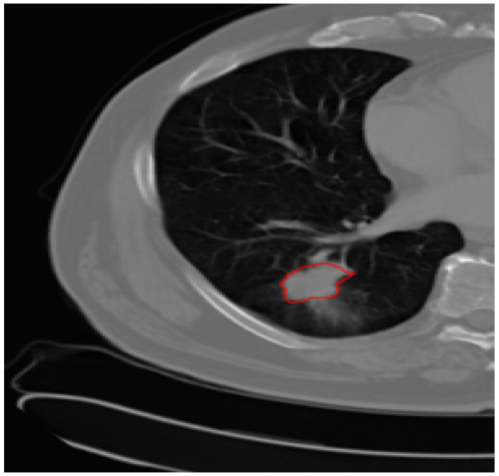
Exact



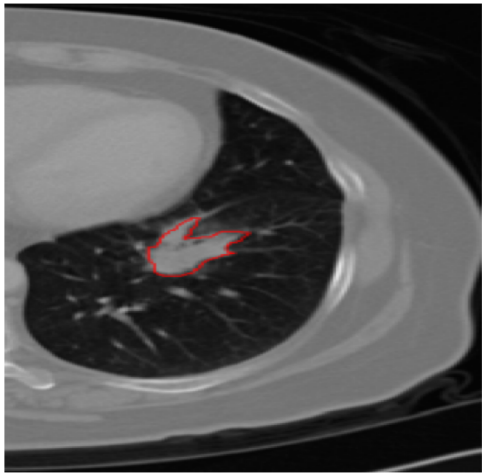
Cropped

If all tumors are the same size, then no warping is necessary,
but you will either miss some tumor or get regions outside the tumor.

Long and Short term survivor Adenocarcionoma images

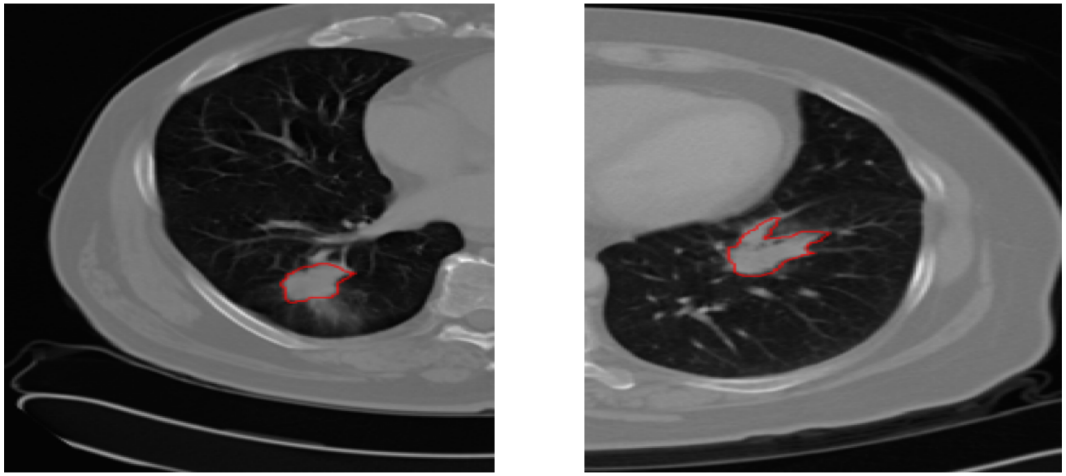


a



b

Long and Short term survivor Adenocarcionoma images



a b

Figure 1. Example CT slices of long term (a) and short term (b) survival groups with tumors outlined.

Transfer Learning (from Deep Neural Networks)

- Use a model trained on data from another domain for your domain
  - Transfer what was learned

- ImageNet deep neural networks are good at classifying non-medical images
  - Can features from them be useful in medical image understanding?

- Features from a deep neural network can be the activations of a hidden layer when presented with a medical image

Addressing issues in Medical Imaging for Deep Learning

- If there are not many images, what about transfer learning from an existing deep neural network?

- If the image size must be fixed, what do we do?
  - Warp all images into the required size
  - Matlab can do it for us
  - It may require a 2D tumor slice

- Can we **tune** an existing trained network with our data?

Addressing issues in Medical Imaging using Deep Learning

- Can we somehow create more data for tuning or training from scratch?
  - Yes! Consider rotating images, some noise on the images, flipping images, etc.
  - Keras can help you create new images

- Generative Adversarial Networks (GAN) may be used (more later)
  - Essentially a game is played between an image recognizer and an image generator trying to fool the recognizer into thinking a generated image is from the original distribution

How much data is available?

- For brain tumor imaging there is a unique set of 22 images of patients that did not undergo resection
  - 13 short survivors and 9 long survivors
- In lung cancer imaging there is a set of 40 Adenocarcinoma cases (20 short and 20 long survivors)
- We created two NLST cohorts each with 85 cancer cases. The first had a total of 261 cases and the second cohort had 237 cases

How much data is available?

- If you think of millions of camera images vs. 261 medical images, there is little data available
- If you have 300,000 weights (MNIST) would you not like 300,000 images?
  - Of course the MNIST accuracy is quite good with 60,000 training examples (on a separate 10,000 test)
- There is no known medical imaging data set (yet) that has even 50,000 images

Augmenting Data

- It would seem that for tuning or building a (small) deep neural network data augmentation is important!

- Almost all medical image evaluation systems doing training or tuning with deep neural networks have used augmented data

Transfer Learning - Example

- Rectified linear units are used to take the output of a node x and map it to max(0,x)
- You can extract features pre or post RELU
- The extracted features can be used in any other classifier, though some check for correlations among features may be in order
- If, for example, extraction is done from an ImageNet trained network you may have up to 4096 features and need to do some feature reduction

Another trained ImageNet Architecture

- CNN – F has 8 learnable layers. An image of 224x224 goes through 5 convolutional layers

Conv1	Conv2	Conv3	Conv4	Conv5
11x11x64 st. 4, pad 0 LRN, x2 pool	5x5x256 st. 1, pad 2 LRN, x2 pool	3x3x256 st. 1, pad 1	3x3x256 st. 1, pad 1	3x3x256 st. 1, pad 1 x2 pool

- Followed by 3 fully connected layers

full6	full7	full8
4096 dropout	4096 dropout	1000 softmax

Transfer Learning

- Ideally, you will tune a trained network with some examples and then use features extracted from it
- How many tuning examples do you need? Open question, but as many as you can obtain is a good answer
- What classes to use in training? How about the ones that most often have the highest activation when you send your data through the network

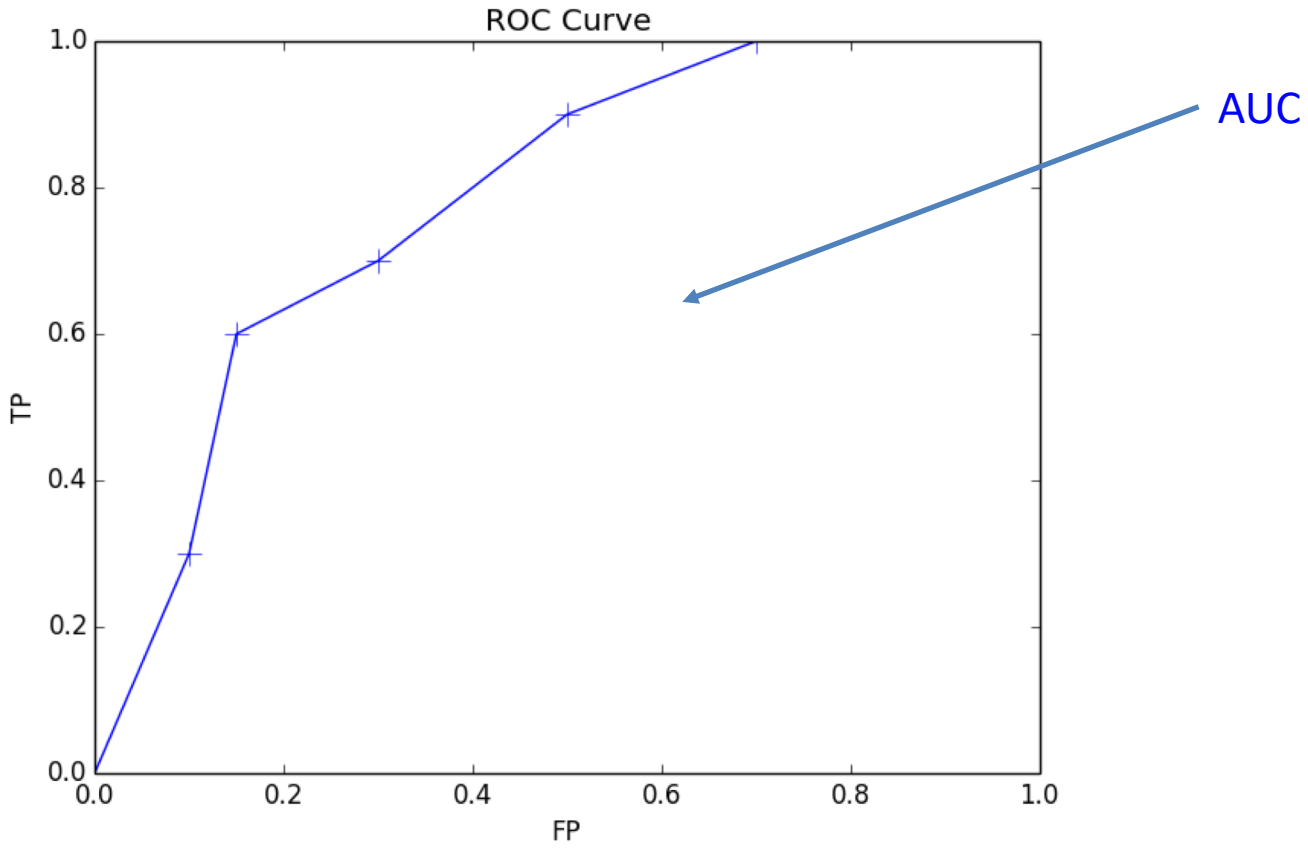
Transfer Learning

- If I use something like ImageNet for a CT or MRI, what channel?
  - One of R, G, B – open question
    - Try all separately?
  - Or should you copy the image and send it through all channels?

Evaluation

- Accuracy – the percentage of predictions correct
- In Medicine the area under the receiver operating characteristic curve, commonly called AUC is popular for comparing classification approaches

ROC Curve from sliding probability threshold



Feature Selection

- Lots of ways to select features, principal components analysis (PCA), Wrappers, and filters

- Here we discuss two classifier independent filter approaches
  - Relief-F looks at nearest neighbors of a randomly chosen example for each feature in order to rank the features
  - Symmetric uncertainty uses mutual information between a feature and a class to rank features

Transfer Learning Example

- 40 Adenocarcinoma images.
  - 20 Stage IV and 20 Stage I/II (Short/Long survival)
  - Can we classify them?

- Send images through the R channel in CNN-F from Matconvnet trained on ImageNet
- Use 56x56 window around the tumor
- Take 4096 features from 7^{th} layer, select 10 with Symmetric Uncertainty
- Accuracy of 77.5% (AUC 0.712) is as good as using 219 carefully selected, extracted features

Transfer Learning

- Without feature extraction and using a trained deep neural network that knows about camera images of scenes
  - The same accuracy as with carefully selected features!

- What were the features?
  - Hard to answer, but important
  - We think texture based, but investigation (explanation) is needed

Combining Deep and Other Features

- We used Relief-F to find the best 5 features from each of the deep neural network and 219 extracted features using an exact window that best covered only the tumor

- With a Naïve Bayes classifier:
  - Accuracy of 90% (AUC 0.935)
  - The increase in AUC was statistically significant at the p=0.05 level.

NLST Experiments

- Two cohorts of patients were created
- Both had nodules that could be tracked across time
- Cohort 1 had a negative nodule that was diagnosed as cancer after 1 year and Cohort 2 had a negative nodule diagnosed as cancer after 2 years
- A model was built from T1 in Cohort 1 to predict the outcome of nodules from T1 in Cohort 2

Cohort 1 details

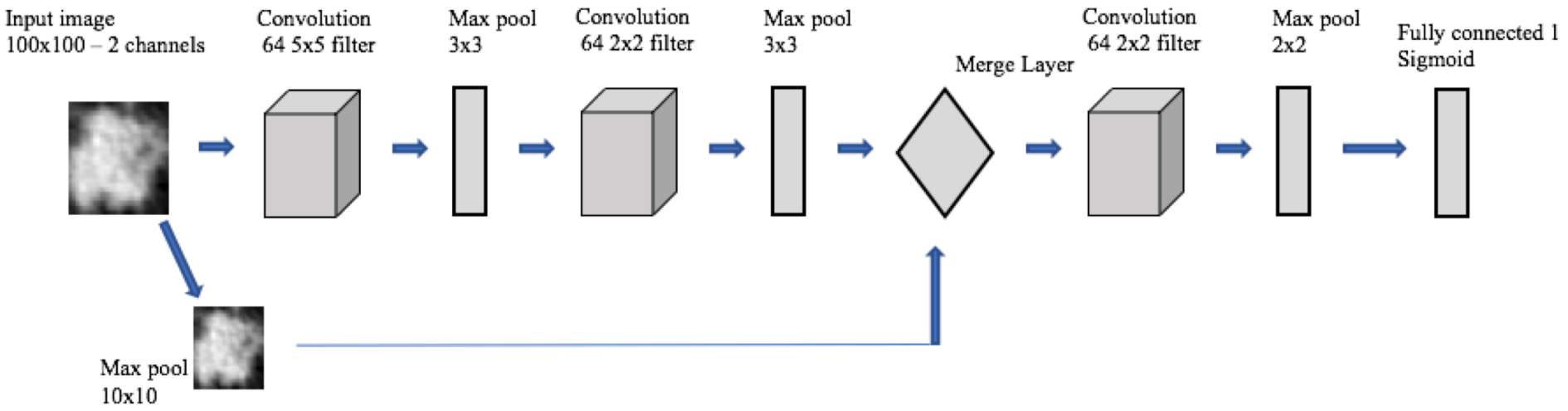
- Cohort 1 had a nodule-positive screen at time T1 and at T2 – approximately 1 year later. A screen-detected lung cancer (SDLC) was diagnosed in 104 of the cohort members at T2
- These individuals were demographically matched to subjects with benign pulmonary nodules (bPNs) and the same screening history. Of the 208 bPNs identified at T1, 176 were successfully segmented along with 85 SDLC

Cohort 2 details

- Cohort 2 had a nodule positive/cancer-negative screen at T1 and T2, followed by a nodule-positive screen at time 3 (T3), with 92 cohort members having screen-detected lung cancer. These individuals were demographically matched to 184 subjects with bPNs, 152 of which were successfully segmented along with 85 SDLC

Results of train on Cohort 1 and test on Cohort 2

- Using our own small CNN trained with augmentations the accuracy was 76% with the highest AUC achieved of 0.87
  - We used 70% of data for training and 30% for validation
  - 15 degree rotation and then horizontal and vertical flipping was done to add 72 images for each real image
  - The architecture used a 100x100 image size in two “channels” with 3 layers of convolutions and only full connections to the output unit
- With transfer learning using all 3 channels of ImageNet, accuracy was 75.1% with 0.78 AUC using random forests on the top 15 features of both as chosen by symmetric uncertainty
- The accuracy is a little less than the 77.64% achieved with 219 texture, size, shape and intensity features. The AUC is the best we have achieved



USF CNN Architecture 3

Ensembles of Deep Neural Networks

- We created two more deep neural networks for the LDCT nodule prediction problem
  - Both a little larger than the one shown previously

- A simple combination of the ensemble of 3 CNN's (average, vote or take the median output) was used

Ensembles of Deep Neural Networks

TABLE 1. CNN architecture 1

Layers	Parameter	Total Parameters
Input Image	100x100	
Conv 1	64 X 5 X 5, pad 0, stride 1	
Leaky ReLU 1	alpha = 0.01	
Max Pool 1	3x3, stride 3, pad 0	
Conv 2	64 X 2 X 2, pad 0, stride 1	
Leaky ReLU 2	alpha = 0.01	841,681
Max Pool 2	3x3, stride 3, pad 0	
Dropout	0.1	
Fully connected 1+ relu	128	
Fully connected 2 + relu	8	
L2 regularizer	0.01	
Dropout	0.25	
Fully connected 3	1 sigmoid	

TABLE 2. CNN architecture 2

Layers	Parameter	Total Parameters
Input Image	100x100	
Conv 1	64 X 5 X 5, pad 0, stride 1	
Leaky ReLU 1	alpha = 0.01	
Max Pool 1	3x3, stride 3, pad 0	
Conv 2	64 X 2 X 2, pad 0, stride 1	
Leaky ReLU 2	alpha = 0.01	845,033
Max Pool 2	3x3, stride 3, pad 0	
Dropout	0.1	
Fully connected 1+relu	128	
LSTM 1 +relu	8	
L2 regularizer	0.01	
Dropout	0.25	
Fully connected 2	1 sigmoid	

Ensembles of Deep Neural Networks

CNN Architecture	Accuracy on Cohort 2	AUC
Arch 1	75.1%	0.82
Arch 2	75.5%	0.86
Arch 3	76%	0.87

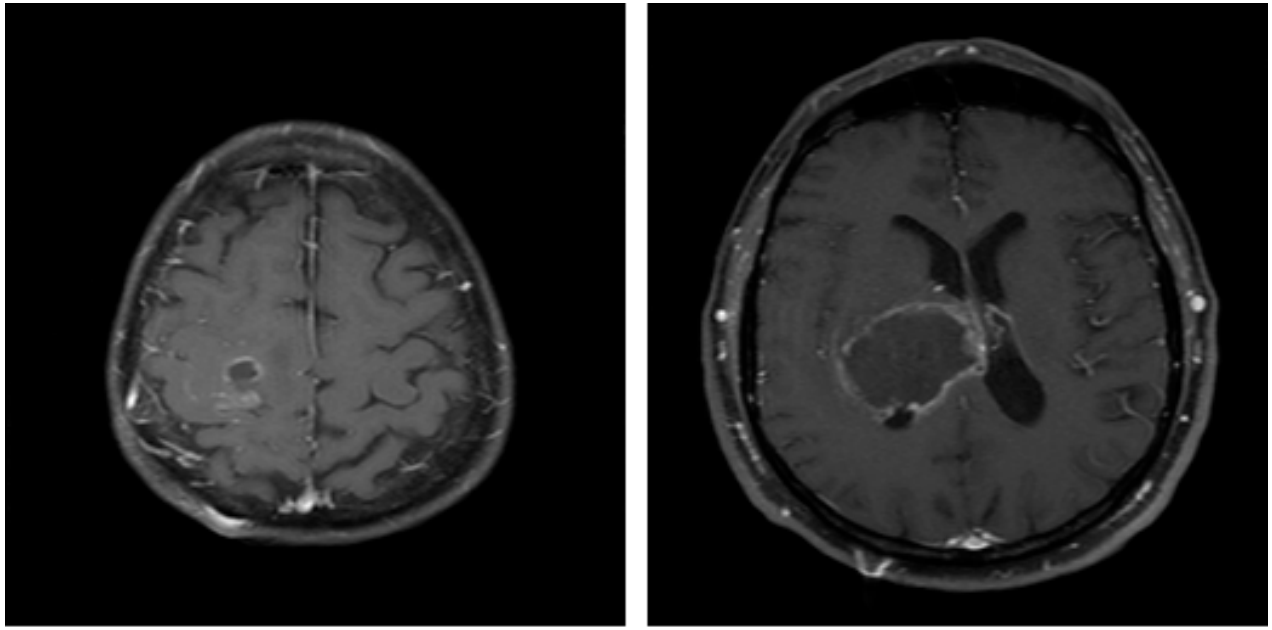
Combination Approach	Average	Median	Vote
Accuracy	86.91%*	80.16%	80.16%
AUC	0.94*	0.91	0.77

* Statistically significant improvement over Arch 3

Brain Tumor Classification

- We have two sets of images of patients with Glioblastoma Multiforme brain tumors.
  - 22 pre and post treatment images of patients who did not have their tumors resected with multiple Magnetic Resonance Image (MRI) sequences
  - 68 patients from the US NIH total cancer image archive with just T1-weighted post contrast sequences
- Long term survival is about a year, 365 days
- These are really SMALL data sets for deep learning
  - Transfer learning?

Brain Tumors



(a) Long term

(b) Short term

Brain Tumor Classification

- Classification is into 2 classes – long term survivor (>365 days) and short term survivor
  - In practice there are many more short term survivors
  - For our data there are 10 long term survivors in the 22 case set and 35 in the 68 case set
- CNN-F pre-trained on ImageNet was used with our 22 example set
- Transfer learning was done taking the outputs of the last fully connected layer as 4096 features

Brain Tumor Classification

- The largest slice of tumor as outlined by a radiologist on the T2-weighted sequence was used from pre-treatment images
- Features were reduced to the top 7 after using symmetric uncertainty, and then Random Forests (200) as the classifier
- A leave one example out cross validation showed the best accuracy was 72.73%.

Brain Tumor Classification

- Next tuning was done using 21 examples in each fold after modifying the network to have just 2 output units

- A learning rate of 0.00001 was used for the first 4 iterations and then it was decreased every 4 iterations by an order of magnitude. We ran the tuning for 10 iterations

- For the FLAIR sequence the accuracy increased from 63.64% to 81.81%

- Still a little less than using LBP features from habitats (86.35%), but impressive with small data and required no feature extraction

Brain Tumor Classification

- Using the 68 publicly available GBM tumors with a T1-sequence with non-deep features the best accuracy was 64% in a LOO CV

- Using an ImageNet model (CNN-F) tuned for 20 iterations and random forests applied to all 4096 features an accuracy of 77.94% after LOO CV was obtained

- Tuning with augmented images did not improve accuracy

How to use DNN's or CNN's

1. Use GPU's for training
2. Start with the architecture of a pre-trained model in Matlab there is matconvnet, http://www.vlfeat.org/matconvnet/
3. Use the above or one of
    1. Convnet: http://torontodeeplearning.github.io/convnet/
    2. Theano: http://deeplearning.net/software/theano/
    3. TensorFlow: https://www.tensorflow.org
    4. Caffe: http://caffe.berkeleyvision.org
    5. TorchL http://torch.ch
    6. Others at: http://deeplearning.net/software_links/

How to use CNN's

- You can start with a pre-trained CNN, say on ImageNet
- ImageNet has 14 million examples in 1000 classes like hammer, house cat, plate, etc.
- Features for your data can be extracted from one of the layers in a CNN trained on ImageNet. We have used the layer before the outputs, but others have been used
- You can tune a pre-trained CNN if you have a fair amount of data
- The last two bullets above have to do with transfer learning

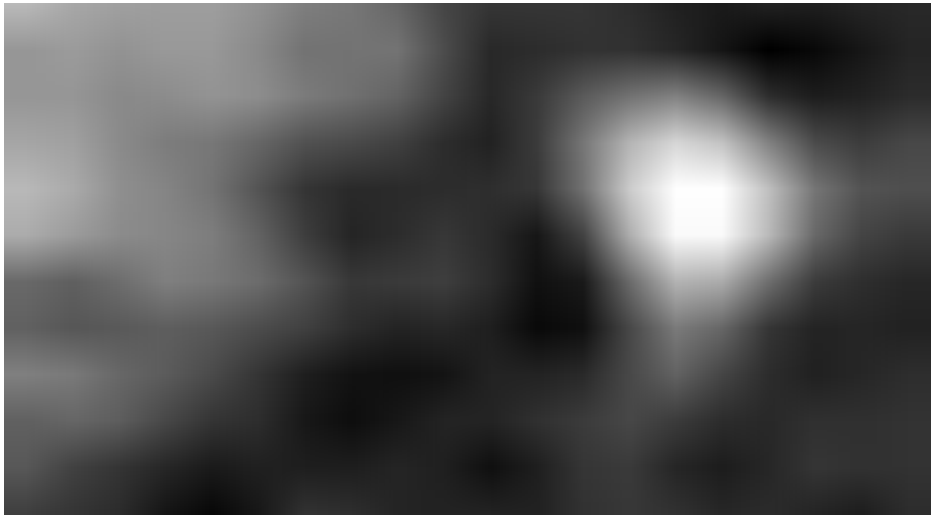
How to use CNN's

- Want to build a CNN or DNN from scratch?
  - Choose your favorite type of CNN/DNN and get access to a GPU machine
  - You would want as close as possible to as many examples as weights (more if possible)
  - An ImageNet size CNN has on the order of 4 million weights and takes weeks (down to days for some architectures) to train with a powerful GPU cluster

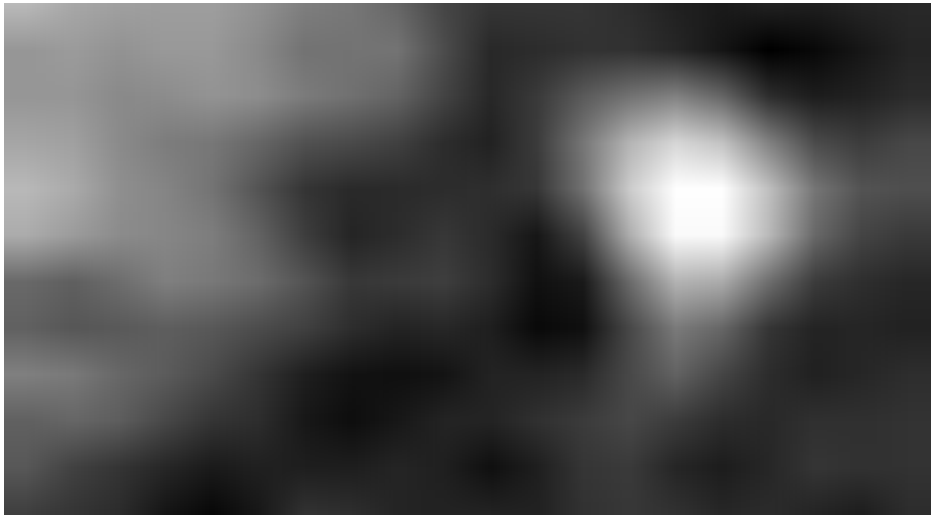
How to use CNN's

- Want to build a CNN or DNN from scratch?
  - You can do some image rotations, flipping, adding noise and other operations to generate more examples
  - You can use generative adversarial networks (GAN's) to generate new images for training (this is not a completely proven approach)
  - In medical imaging 4M examples seems a stretch, but you can try smaller networks (smaller numbers of images). We have seen smaller networks can be pretty good

Real or GAN Nodules



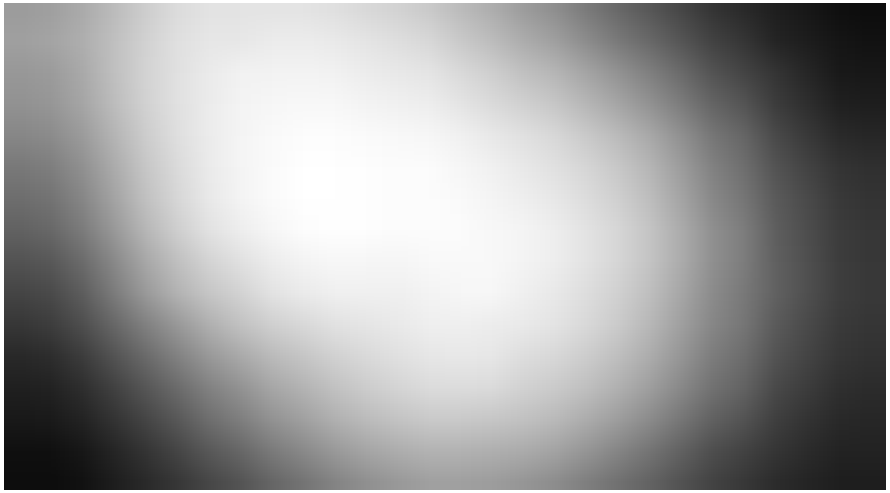
Real or GAN Nodules



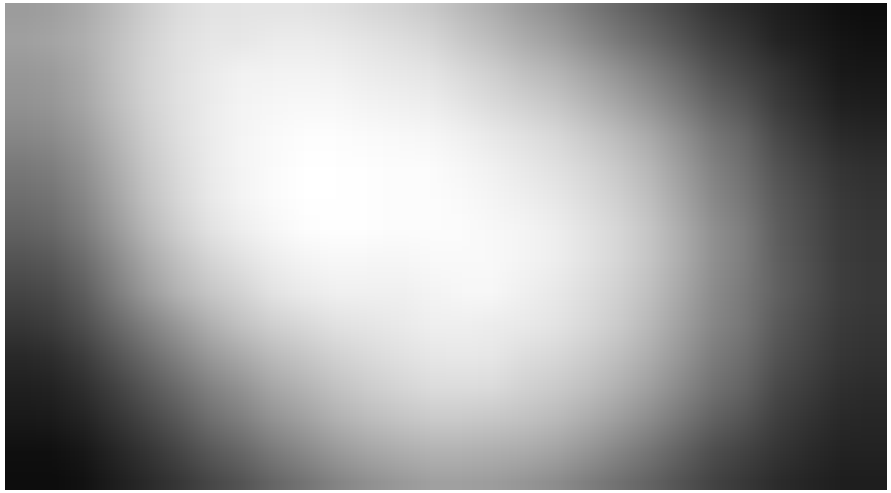
GAN

REAL

Real or GAN Nodules



Real or GAN Nodules



REAL

GAN

How to use CNN's for medical image analysis

- U-Net enables segmentation of images, https://arxiv.org/abs/1505.04597
- For medical images someone has to segment tumors or abnormalities
- You can train a U-Net or some other deep network to do this
  - It was pretty successful in a lung nodule/tumor differentiation challenge on Kaggle

Summary

- Deep Neural Networks, with tweaks, can and have been used with success on small datasets
  - Dermascopy provides a medium size data set where results have been good
  - U-Net learns to segment with relatively small numbers of examples
- Transfer learning works surprisingly well on image data
- Deep Neural Networks are “pretty dominant” in accuracy with lots of labeled training data

Questions?

References

[1] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3547{3555, June 2015.

[2] K. Chateld, K. Simonyan, A. Vedaldi, and Andrew A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531, 2014.

[3] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classication. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3642{3649. IEEE, 2012.

[4] George E Dahl, Tara N Sainath, and Georey E Hinton. Improving deep neural networks for lvcsr using rectied linear units and dropout. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8609{8613. IEEE, 2013.

[5] Je Donahue, Yangqing Jia, Oriol Vinyals, Judy Homan, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML, pages 647{655, 2014.

References

[6] Denton E.L., Chintala S., and Fergus R. et. al. Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems, pages 1486{1494, 2015.

[7] M. Ghafoorian, N. Karssemeijer, T. Heskesy, I.W.M. van Uden, F.E. de Leeuw, E. Marchioriy, B. van Ginneken, and B. Platel. Non-uniform patch sampling with deep convolutional neural networks for white matter hyperintensity segmentation. In IEEE International Symposium on Biomedical Imaging, 2016.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on articial intelligence and statistics, pages 249{256, 2010.

[9] Alex Graves and J?urgen Schmidhuber. Oine handwriting recognition with multidimensional recurrent neural networks. In Advances in neural information processing systems, pages 545-552, 2009.

[10] J. Kawahara, A. BenTaieb, and G. Hamarneh. Deep features to classify skin lesions. In 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI), pages 1397{1400, April 2016.

[11] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classication with deep convolutional neural networks. In Advances in Neural Infor- mation Processing Systems, pages 1097{1105, 2012.

[12] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. The Journal of Machine Learning Research, 10:1{40, 2009.

[13] R. Liu, L. O. Hall, D. B. Goldgof, M. Zhou, R. A. Gatenby, and K. B. Ahmed. Exploring deep features from brain tumor magnetic resonance images via transfer learning. In International Joint Conference on Neural Networks, 2016.

[14] M. Mishra, S. Schmitt, L. Wang, M. K. Strasser, C. Marr, N. Navab, H. Zischka, and F. Tingying. Structure-based assessment of cancerous mitochondria using deep networks. In IEEE International Symposium on Biomedical Imaging, 2016.

[15] Rahul Paul, Samuel H. Hawkins, Y. Balagurunathan, Matthew B. Schabath, Robert J. Gillies, Lawrence O. Hall, and Dmitry B. Goldgof. Deep feature transfer learning in combination with traditional features predicts survival among patients with lung adenocarcinoma. Tomography, 2(4):388{395, December 2016.

References

[16] Geethika Bhavya Peddibhotla and KDnuggets. 50 deep learning software tools and platforms, updated, http://www.kdnuggets.com/2015/12/deep-learning-tools.html.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Neural Information Processing Systems (NIPS), 2015.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1{9, 2015.

[19] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Transactions on Medical Imaging, 35(5):1299{1312, May 2016.

[20] I. H. Witten, E. Frank, M. Hall, and C.J. Pal. Data Mining: Practical machine learning tools and Techniques. Morgan Kaufmann, San Francisco, fourth edition, 2017.

[21] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems, pages 3320{3328, 2014.

Contest with MNIST data

- You form groups of 3, make allowed changes to model, I test the outputs
- Winning team gets a prize!
- Default Architecture: 5x5x32, maxpool 2x2, 5x5x64, maxpool 2x2, then 7x7 feature maps to 1024 hidden units to 10 output units

Contest with MNIST data – Cannot Change

- Number of layers or general functionality of them
- Just 10,000 cycles through batches allowed
- Cannot change loss function from cross_entropy

Contest with MNIST data – Can Change

- Validation set size: currently 7500
- Can change the kernel – Currently 5x5 with padding on both sides, stride of 1 for conv. Layers. Only kernel size for 1 or more layers
- Can change from RELU to leaky_relu or sigmoid: any layer(s)

Contest with MNIST data – Can Change

- Can change batch size from default of 24. Can change batch size once after n batches processed
- Can change dropout probability from 0.5
- Can change learning rate from 1xe-4
- Can change fully connected layer from 1024 units

Electronic Entry Form

- http://www.cse.usf.edu/~lohall/MNIST%20CNN%20Parameters%20TeamEntry.docx
- Or
- http://www.cse.usf.edu/~lohall/MNISTCNNParametersTeamEntry.pdf