Lecture 20: Introduction to Deep Learning

- Basics of feedforward Neural Networks
- Back-Propagation
- Convolutional Neural Net
- Some design examples

Logistic Regression and Logistic Function

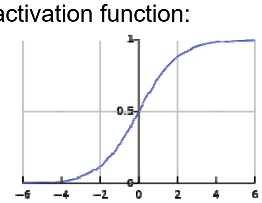
Logistic regression has a learned parameter vector W. On input X, label $Y \in \{-1,1\}$, the logistic regression is defined as generalized linear model with link function:

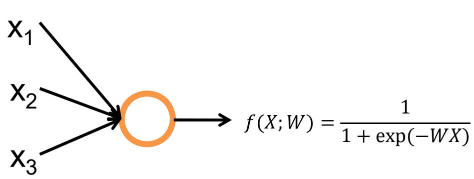
$$\log \frac{P(Y=1|X)}{P(Y=-1|X)} = WX$$

$$P(Y=1|X) = \frac{1}{1+\exp(-WX)}$$

In early days, neural networks use logistic function as activation function:

$$f(X;W) = \frac{1}{1+\exp(-WX)}$$



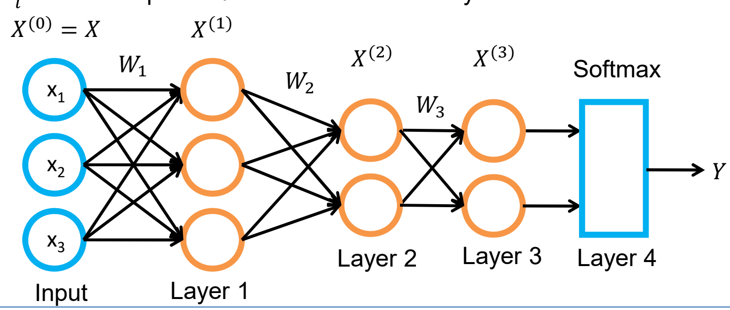


Neural Network

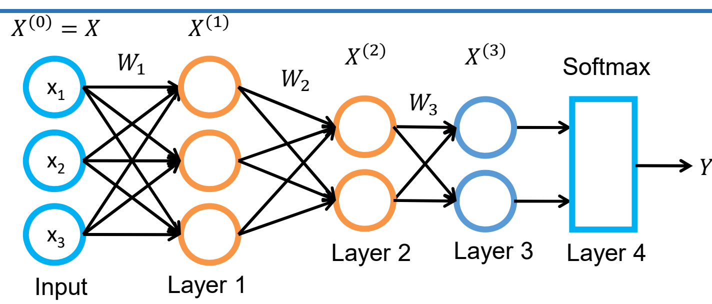
Example: 3 layer network with classification:
In multi-class classification, label $Y \in \{1,2,\cdots,n\}$ has n classes. In modern deep learning, softmax layer as multinomial logistic regression is used to compute the probability of each class, and output $Y = \underset{Y}{\operatorname{argmax}} P(Y|X)$

$$P(Y=i|X) = \frac{\exp(X_i^{(3)})}{\sum_{j=1}^{n} \exp(X_j^{(3)})}, i = 1,2\cdots,n$$

Where $X_i^{(3)}$ is the output for i-th class in the third layer.



Training a Neural Network



Given training set $(X_1, Y_1), (X_2, Y_2), \cdots, (X_M, Y_M), \cdots$, adjust parameters W (for every node) to maximize likelihood:

$$P(Y=i|X;W) = \frac{\exp(X_i^{(L)})}{\sum_{j=1}^{n} \exp(X_j^{(L)})}$$

$X^{(L)}$= <W, $X^{(L-1)}$>: final layer.
$X^{(l)}$: output from l-th layer, l=1,2, ..., $L-1$.
$X^{(0)} = X$ is the input data.

Use gradient ascent. "Back-propagation" algorithm.

In 1980's, the reconstruction error is used as loss function $\sum_{m=1}^{M} \left(Y_m - f(X_m)\right)^2$

Back-propagation

Let $X^{(l)}$ denote the l-th layer output, $l = 1, \cdots, l, \cdots, L$

The gradient for l-th layer parameter:

Chain rule

$$\Delta W_l^t = \frac{\partial P(Y=i|X;W^t)}{\partial W_l^t} = \frac{\partial P(Y=i|X;W^t)}{\partial X^{(L)}} \cdot \boxed{\frac{\partial X^{(L)}}{\partial X^{(L-1)}} \cdots \frac{\partial X^{(l+1)}}{\partial W_l^t}}$$

$$\frac{\partial P(Y=i|X;W^t)}{\partial X^{(L)}} = [p_1 \cdot (1_{(1=i)} - p_i), \cdots, p_k \cdot (1_{(k=i)} - p_i), \cdots, p_n \cdot (1_{(n=i)} - p_i)]$$

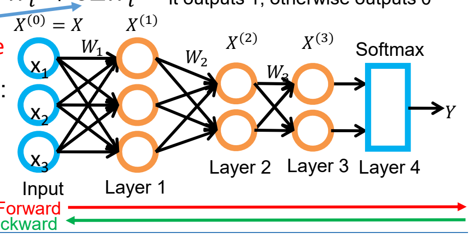
$$p_k = \frac{\exp(X_k^{(L)})}{\sum_{j=1}^{n} \exp(X_k^{(L)})}, k = 1,2,\cdots,n$$

$1_{(k=i)}$ is Indicator function, if $k = i$, it outputs 1, otherwise outputs 0

$$W_l^{t+1} = W_l^t + \delta \Delta W_l^t$$

δ : small learning rate

Learning algorithm involves two passes:
1) Forward: Compute p_k
2) Backward: Compute gradient and update parameter W



Back-propagation

Chain rule part:

Let $X^{(l)}$ denote the l-th layer output, $l = 1, \cdots, l, \cdots, L$

$$X^{(l)} = f_l(X^{(l-1)}; W_l) = \frac{1}{1 + \exp(-W_l X^{(l-1)})}$$

$$\frac{\partial X^{(l)}}{\partial X^{(l-1)}} = \frac{1}{1 + \exp(-W_l X^{(l-1)})} \frac{\exp(-W_l X^{(l-1)})}{1 + \exp(-W_l X^{(l-1)})} W_l$$

$$\frac{\partial X^{(l)}}{\partial W_l} = \frac{1}{1 + \exp(-W_l X^{(l-1)})} \frac{\exp(-W_l X^{(l-1)})}{1 + \exp(-W_l X^{(l-1)})} X^{(l-1)}$$

The gradient in deep learning is unstable, tending to either explode (large gradient) or vanish (small gradient) in early layers. Bach-normalization is used to normalize output of each layer with a mean of 0 and a standard deviation of 1.

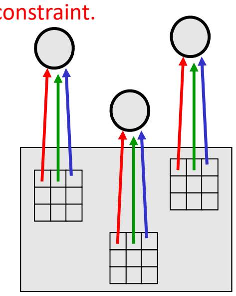
Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536.

Convolutional Neural Network

The replicated feature approach.

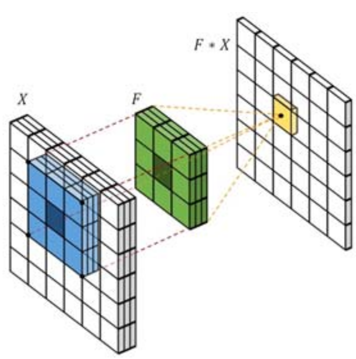
- Use many different copies of the same feature detector with different positions.
  - Could also replicate across scale and orientation (tricky and expensive)
  - Replication greatly reduces the number of free parameters to be learned.
- Use several different feature types, each with its own map of replicated detectors.
  - Allows each patch of image to be represented in several ways.

The red connections all have the same weight with constraint.



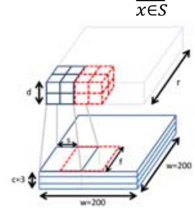
LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition". Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791.

Convolution



Let F denote the filter, the convolution is defined:

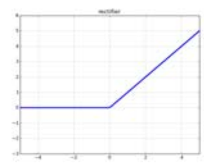
$$[F * X](y) = \sum_{x \in S} W_x X(y + x)$$



The convolution layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume.

Nonlinear Transformation

Rectified linear units (ReLU):
the rectifier is an activation function defined as:



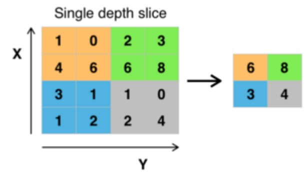
$$f(X) = \max(X, 0)$$

Motivated by strong biological motivations and mathematical justifications

It has been widely used and the most popular activation function in ConvNet.

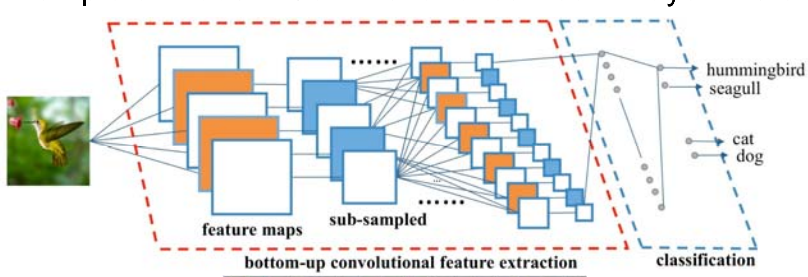
R Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, H.S. Seung (2000). *Digital selection and analogue amplification coesist in a cortex-inspired silicon circuit*. *Nature*. **405**. pp. 947–951.

Max pooling:
Max pooling is a form of non-linear down-sampling. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.



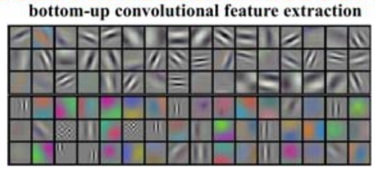
Convolutional Neural Network

Example of modern ConvNet and learned 1st layer filters:



1,000 categories

1st layer learned filters



Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks". Advances in Neural Information Processing Systems. 1: 1097–1105.

Convolutional Neural Network

Formally, a Convolutional Neural Network (ConvNet) is used to define a mapping:

$$Y = f(X;\, W)$$

where W denotes the parameters. $f(\cdot)$ is a composition of L layers of liner transformations followed by element-wise non-linear transformations:

$$X^{(l)} = f_l(W_l X^{(l-1)})$$

where $l = 1, \cdots, L$, $X^{(0)} = X, X^{(L)} = Y$, W_l is the weights matrix. f_l is element-wise nonlinear transformation.

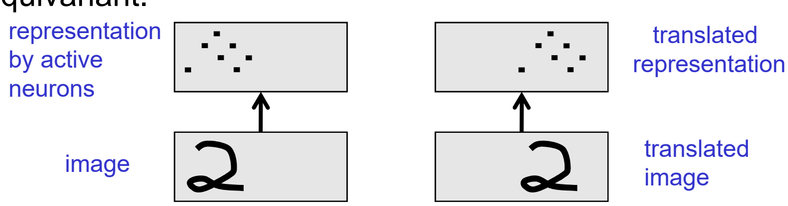
Key factors why modern ConvNets work (empiricle observations)

- ReLU activation function.
- Multinomial logistic regression
- Fully connected layer
- Dropout (Remove 50% of connection)

Hinton, Geoffrey E.; Srivastava, Nitish; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan R "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Jmlr.org. Retrieved July 26, 2015.

What does replicating the feature detectors achieve?

- Equivariant activities: Replicated features do not make the neural activities invariant to translation. The activities are equivariant.



- Invariant knowledge: If a feature is useful in some locations during training, detectors for that feature will be available in all locations during testing.

Pooling the outputs of replicated feature detectors

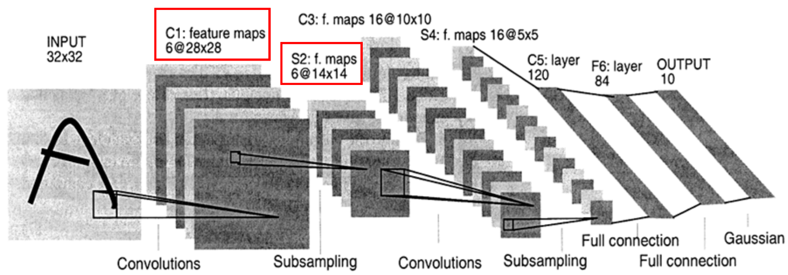
- Get a small amount of translational invariance at each level by averaging four neighboring replicated detectors to give a single output to the next level.
  - This reduces the number of inputs to the next layer of feature extraction, thus allowing us to have many more different feature maps.
  - Taking the maximum of the four works slightly better.
- Problem: After several levels of pooling, we have lost information about the precise positions of things.
  - This makes it impossible to use the precise spatial relationships between high-level parts for recognition.

LeNet

- Yann LeCun and his collaborators developed a really good recognizer for handwritten digits by using backpropagation in a feedforward net with:
  - Many hidden layers
  - Many maps of replicated units in each layer.
  - Pooling of the outputs of nearby replicated units.
  - A wide net that can cope with several characters at once even if they overlap.
  - A clever way of training a complete system, not just a recognizer.
- This net was used for reading ~10% of the checks in North America.

LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition". Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791.

The architecture of LeNet5



Handwriting Digital Recognition

The 82 errors made by LeNet5

Notice that most of the errors are cases that people find quite easy.

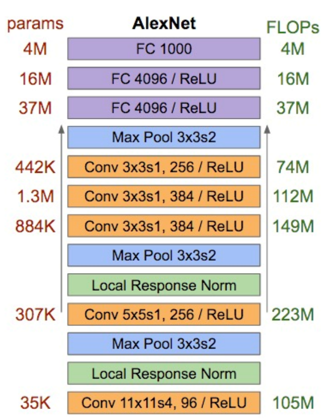
The human error rate is probably 20 to 30 errors but nobody has had the patience to measure it.

Priors and Prejudice

- We can put our prior knowledge about the task into the network by designing appropriate:
  - Connectivity.
  - Weight constraints.
  - Neuron activation functions
- This is less intrusive than hand-designing the features.
  - But it still prejudices the network towards the particular way of solving the problem that we had in mind.
- Alternatively, we can use our prior knowledge to create a whole lot more training data.
  - This may require a lot of work (Hofman&Tresp, 1993)
  - It may make learning take much longer.
- It allows optimization to discover clever ways of using the multi-layer network that we did not think of.
  - And we may never fully understand how it does it.

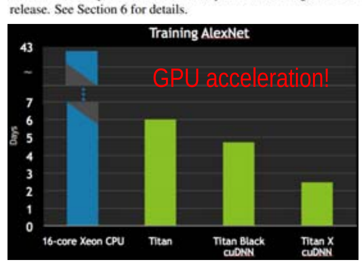
Hinton, Geoffrey. Notes for ConvNet.

AlexNet



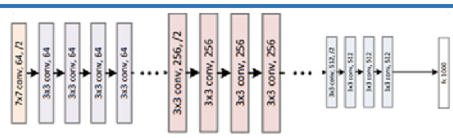
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.



Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks". Advances in Neural Information Processing Systems. 1: 1097–1105.

Deep Learning on ImageNet

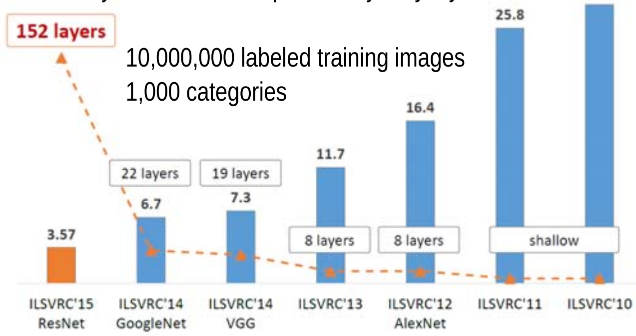


152 Layer and more than 200 MB file size!!

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition, CVPR, 2016.

ConvNets recursively defines filter responses layer by layer.

10,000,000 labeled training images
1,000 categories



ImageNet Classification top-5 error (%)