http://inst.eecs.berkeley.edu/~ee290-2/sp21/

Hardware for Machine Learning

Lecture 3: DNNs 2

Sophia Shao

Apple buys edge-based AI startup Xnor.ai for a reported $200M

January, 15, 2020

Xnor.ai, spun off in 2017 from the nonprofit Allen Institute for AI (AI2), has been acquired by Apple for about $200 million. Xnor.ai began as a process for making machine learning algorithms highly efficient — so efficient that they could run on even the lowest tier of hardware out there, things like embedded electronics in security cameras that use only a modicum of power.

https://techcrunch.com/2020/01/15/apple-buys-edge-based-ai-startup-xnor-ai-for-a-reported-200m

79v4 [cs.CV] 2 Aug 2016

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

Mohammad Rastegari†, Vicente Ordonez†, Joseph Redmon*, Ali Farhadi†*

Allen Institute for AI†, University of Washington*
{mohammadr,vicenteor}@allenai.org
{pjreddie,ali}@cs.washington.edu

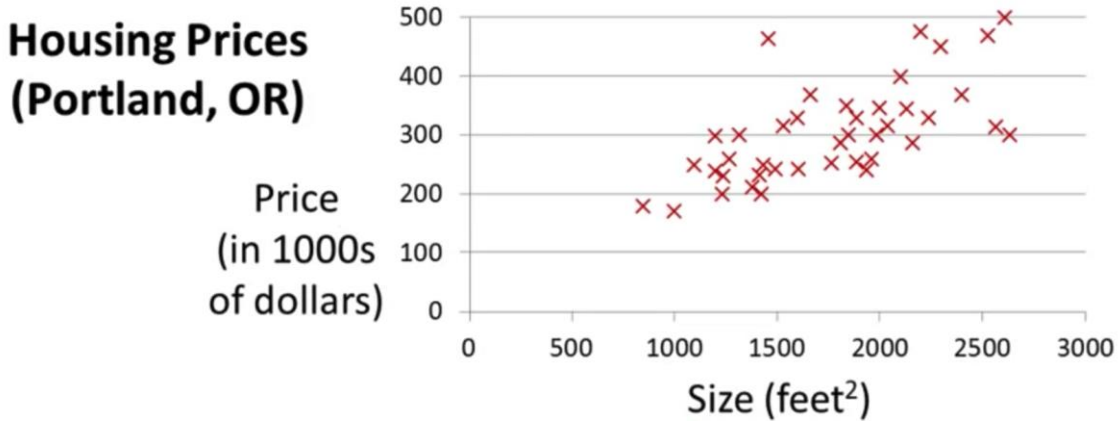
Abstract. We propose two efficient approximations to standard convolutional neural networks: Binary-Weight-Networks and XNOR-Networks. In Binary-Weight-Networks, the filters are approximated with binary values resulting in 32× memory saving. In XNOR-Networks, both the filters and the input to convolutional layers are binary. XNOR-Networks approximate convolutions using primarily binary operations. This results in 58× faster convolutional operations (in terms of number of the high precision operations) and 32× memory savings. XNOR-Nets

Review

- Artificial intelligence, machine learning, and deep learning
- Building a machine learning algorithm:
  - Dataset
  - Cost function
  - Optimization function
  - Model
- Deep learning to automatically extract hierarchical data
  - Better at handle high-dimensional data
  - Key differences are in the Model
    - Multiple layers, i.e., deep

Example: Linear Regression

- Dataset:
  - (x, y) where x is size and y is price
  - m training examples
- Cost function:
  - Mean Squared error
  - $MSE = \frac{1}{m}\sum_{i=0}^{m}(h(x_i) - y_i)^2$
- Model:
  - $h = w_0 + w_1 * x$
- Optimization method:
  - Solve for where its gradient is 0.
  - Gradient descent

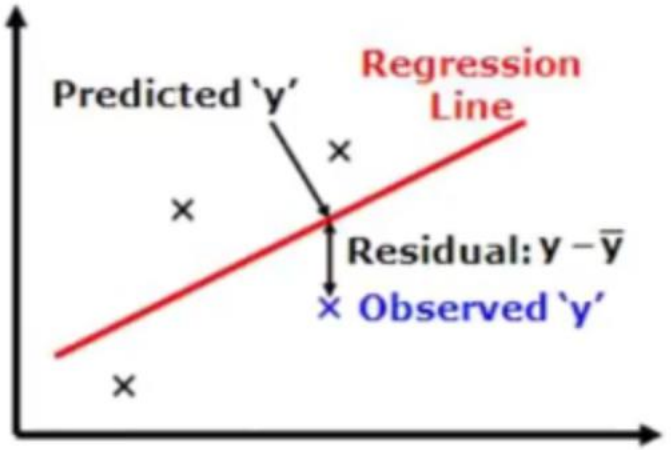


Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Machine Learning, Andrew Ng

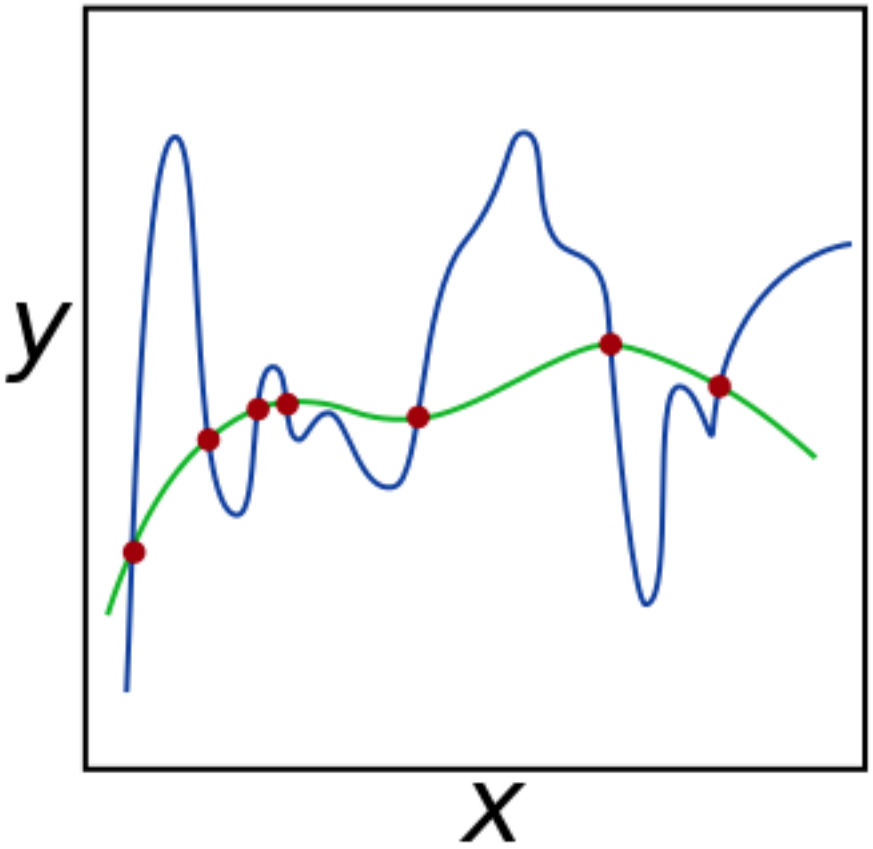
Cost function

- Measures the performance of a ML model for given data
- Quantifies the error between predicted and expected values in the ***training*** set.
  - $CF = MSE_{train} = \frac{1}{m}\sum_{i=0}^{m}(\hat{y}_i - y_i)^2$



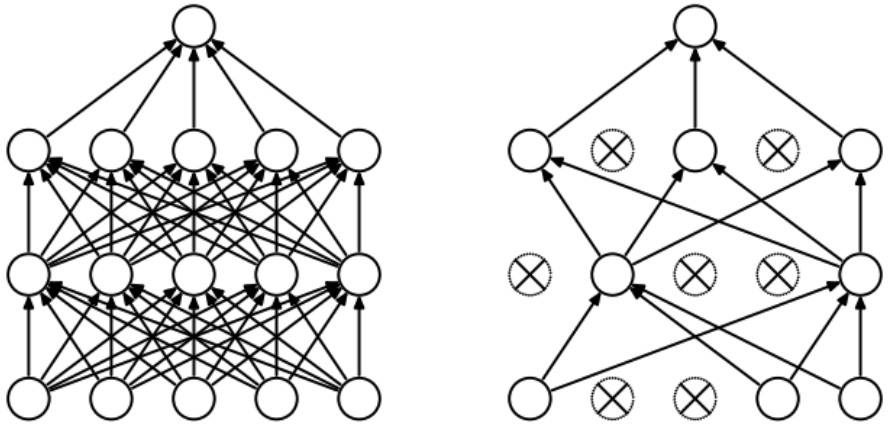
Cost function

- Regularization: modifications to reduce the generalization error but not the training errors
  - Weight decay (L2/1 regularization)
    - Expressing preferences of smaller weights
    - $CF = MSE_{train} + \lambda \sum_i w_i^2 \ (L2)$
    - $CF = MSE_{train} + \lambda \sum_i |w_i| \ (L1)$



Cost function

- Regularization: modifications to reduce the generalization error but not the training errors
  - Dropout
    - Temporally remove nodes from network
    - Train a large ensemble of models that share parameters



(a) Standard Neural Net (b) After applying dropout.

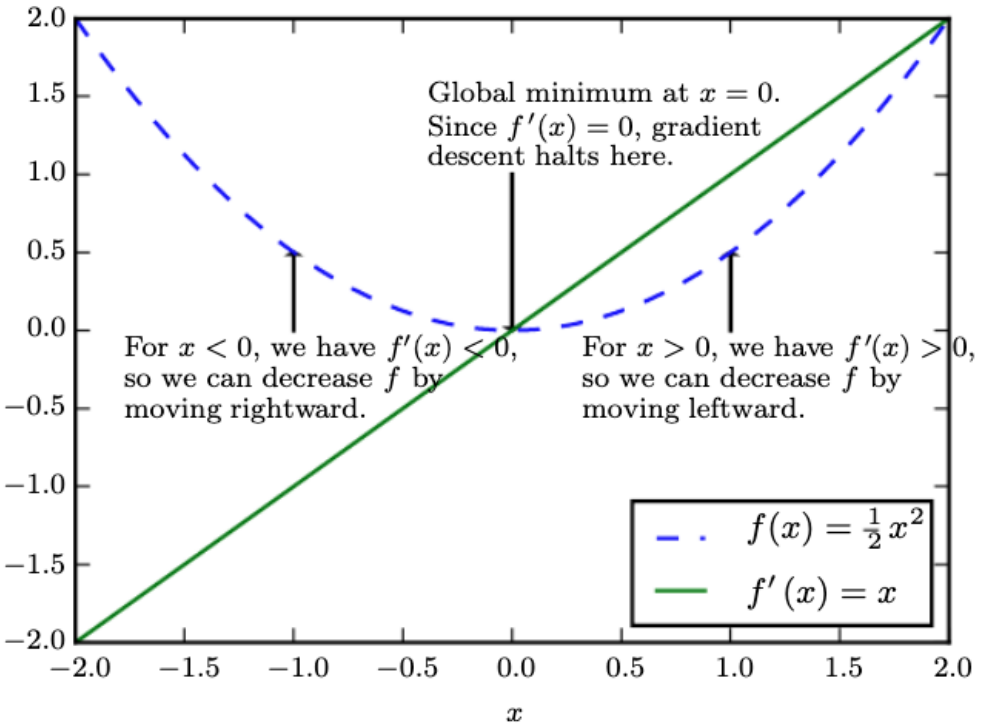
Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Optimization

- Follow the slope

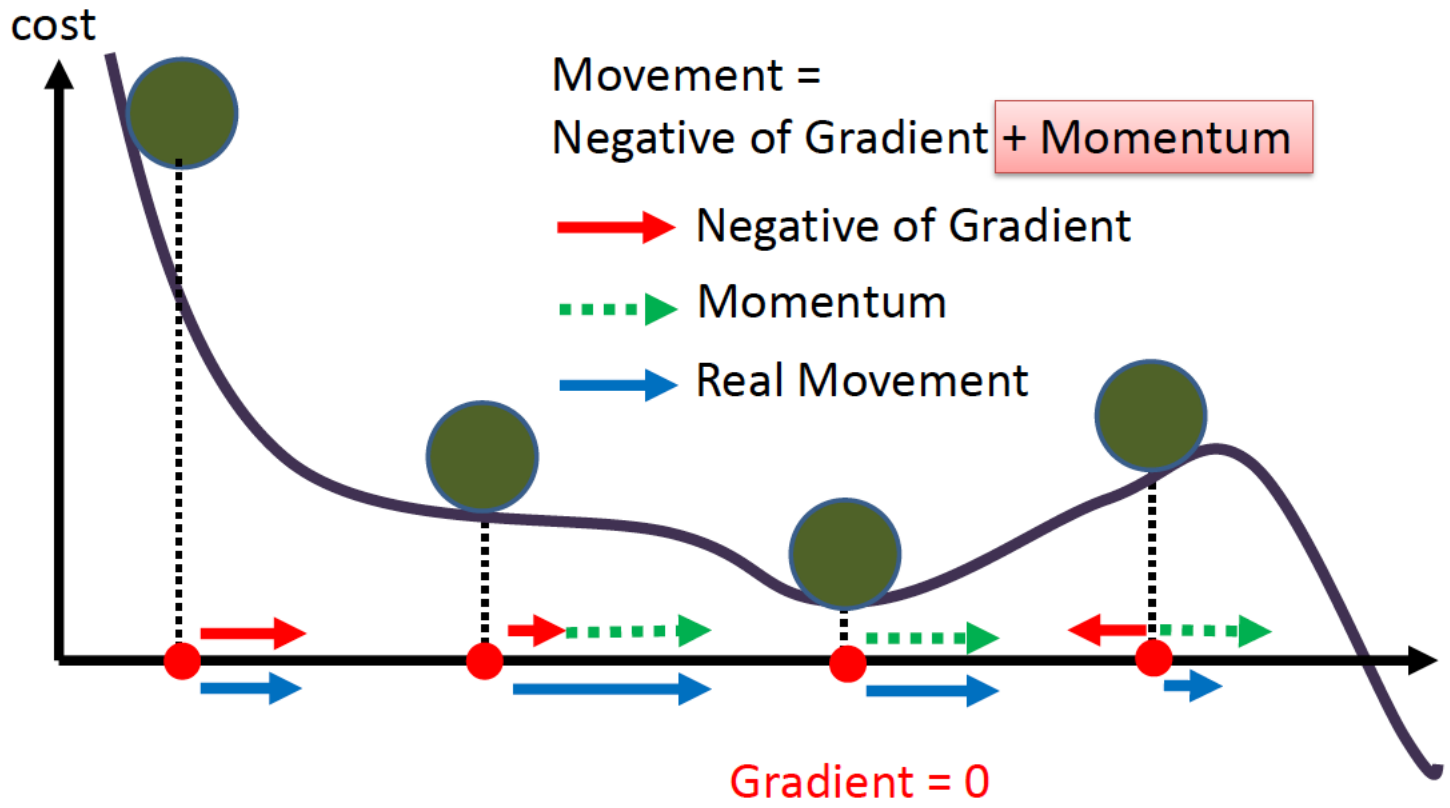
Optimization

- Method to minimize the ***cost function*** by updating weights
- Gradient descent:
  - Iteratively moving in the direction of steepest descent as defined by the negative of the gradient
- Stochastic gradient descent (SGD)
  - To handle large training sets
  - Only run a subset of the training sets (i.e., batch/minibatch) for each update
  - Easier to converge



Optimization

- Momentum:
  - Prefers to go in a similar direction as before



SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
while True:
  dx = compute_gradient(x)
  x -= learning_rate * dx
```

SGD+Momentum

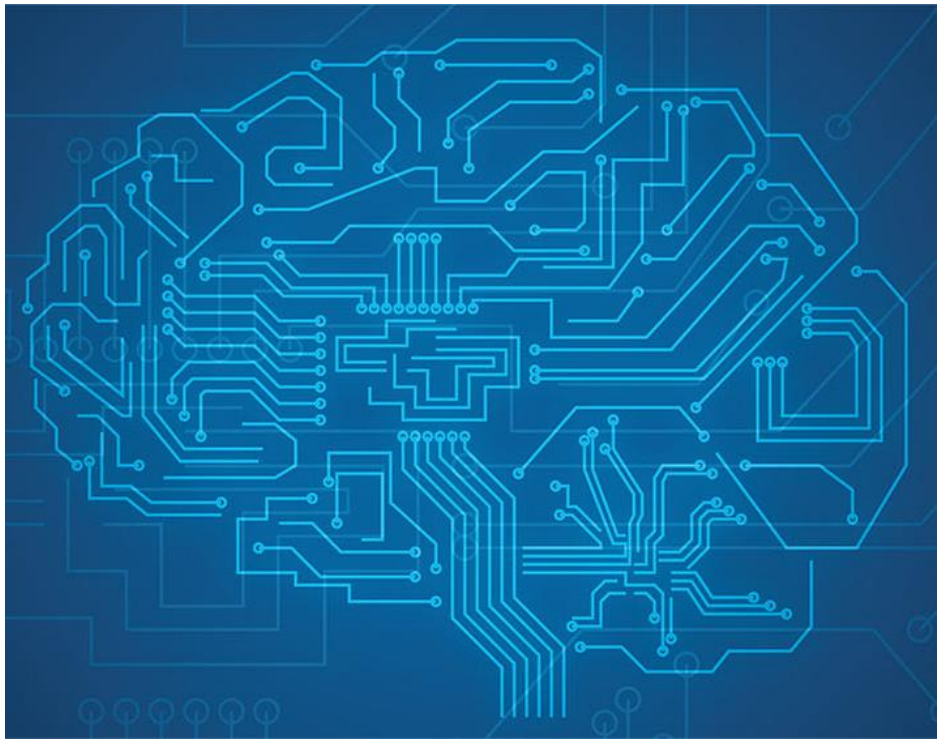
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
  dx = compute_gradient(x)
  vx = rho * vx + dx
  x -= learning_rate * vx
```

Challenges motivating DL

- The curse of dimensionality
  - Generalizing to new examples become exponentially more difficult when working with high-dimensional data
  - Challenging to learn complicated functions in high-dimensional spaces

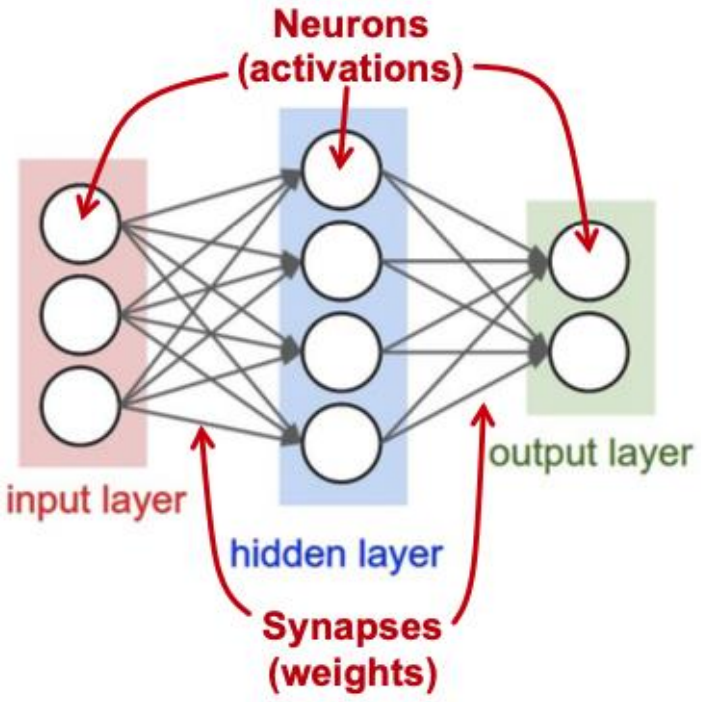


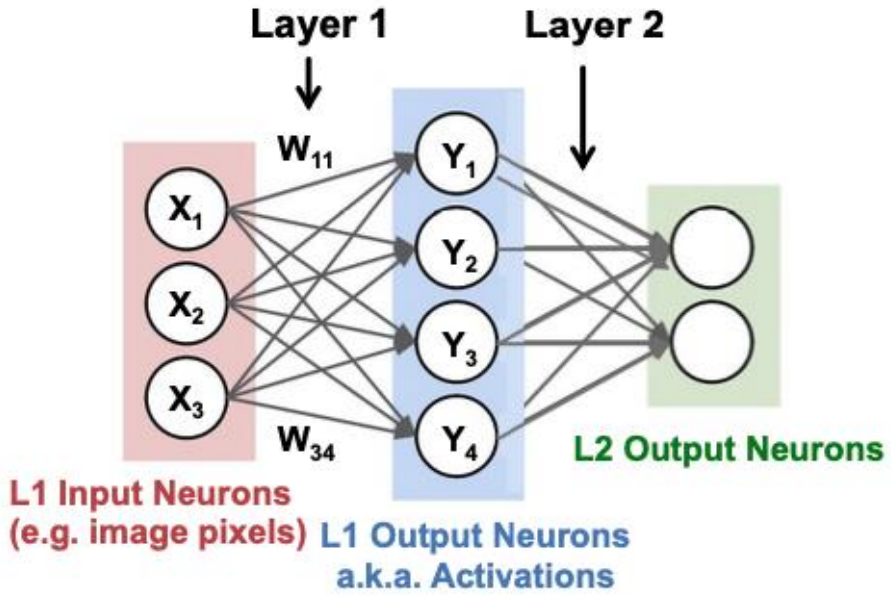
DNNs

- AI, ML, and DL
- ML Basics
- **DL Overview**
- AlexNet Example

Deep Neural Networks

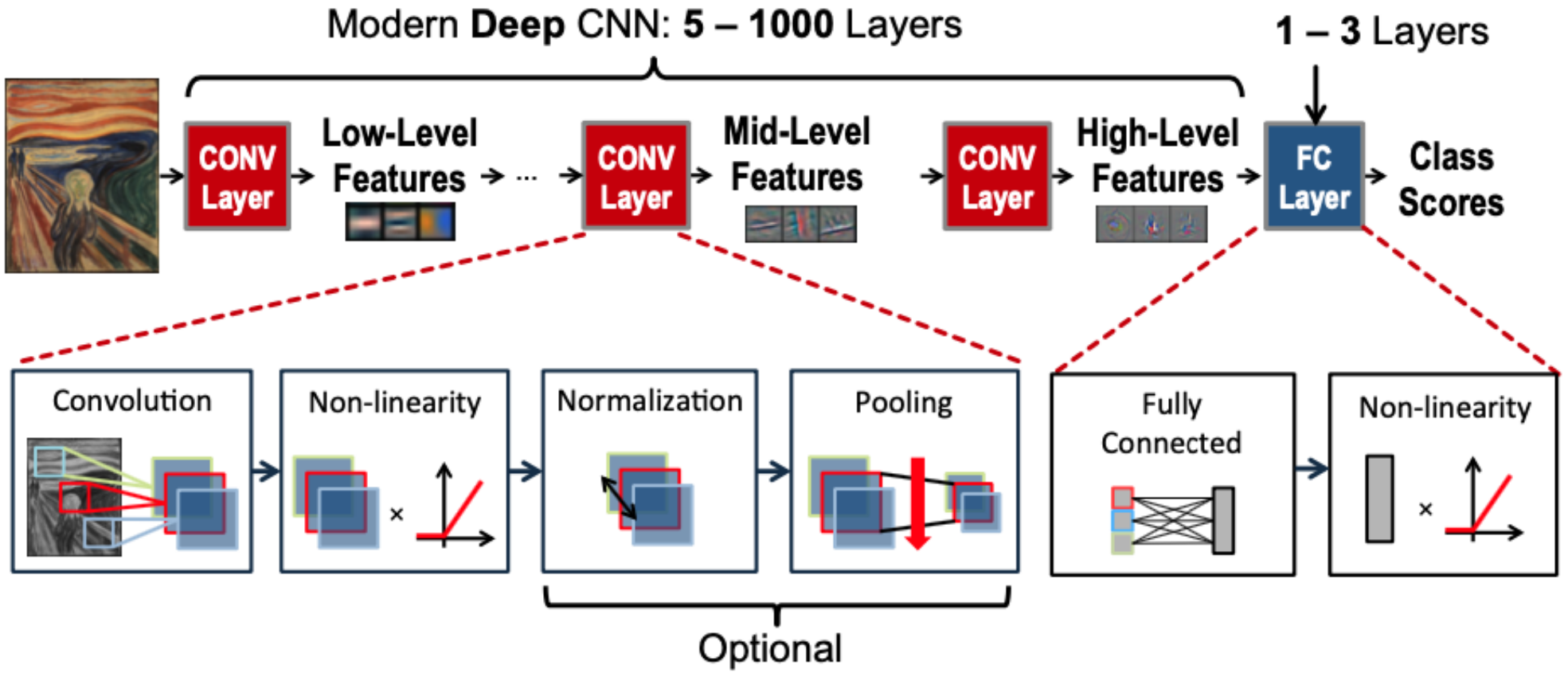
- A subset of machine learning models
  - Deep -> multiple layers
  - Linear model + nonlinear transformation





Eyeriss tutorial

Deep Neural Networks



Eyeriss tutorial

Deep Neural Networks: LeNet

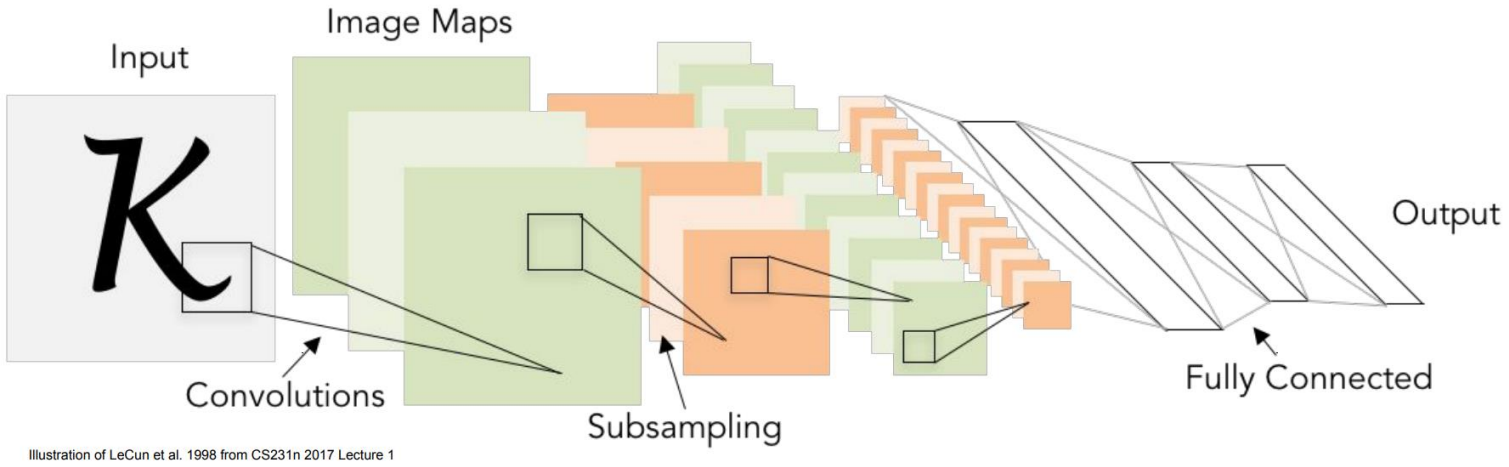
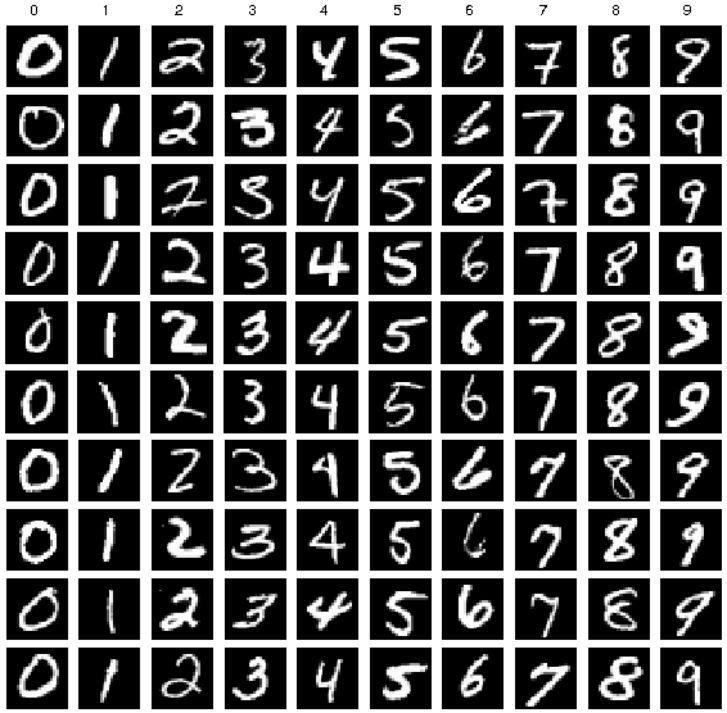


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

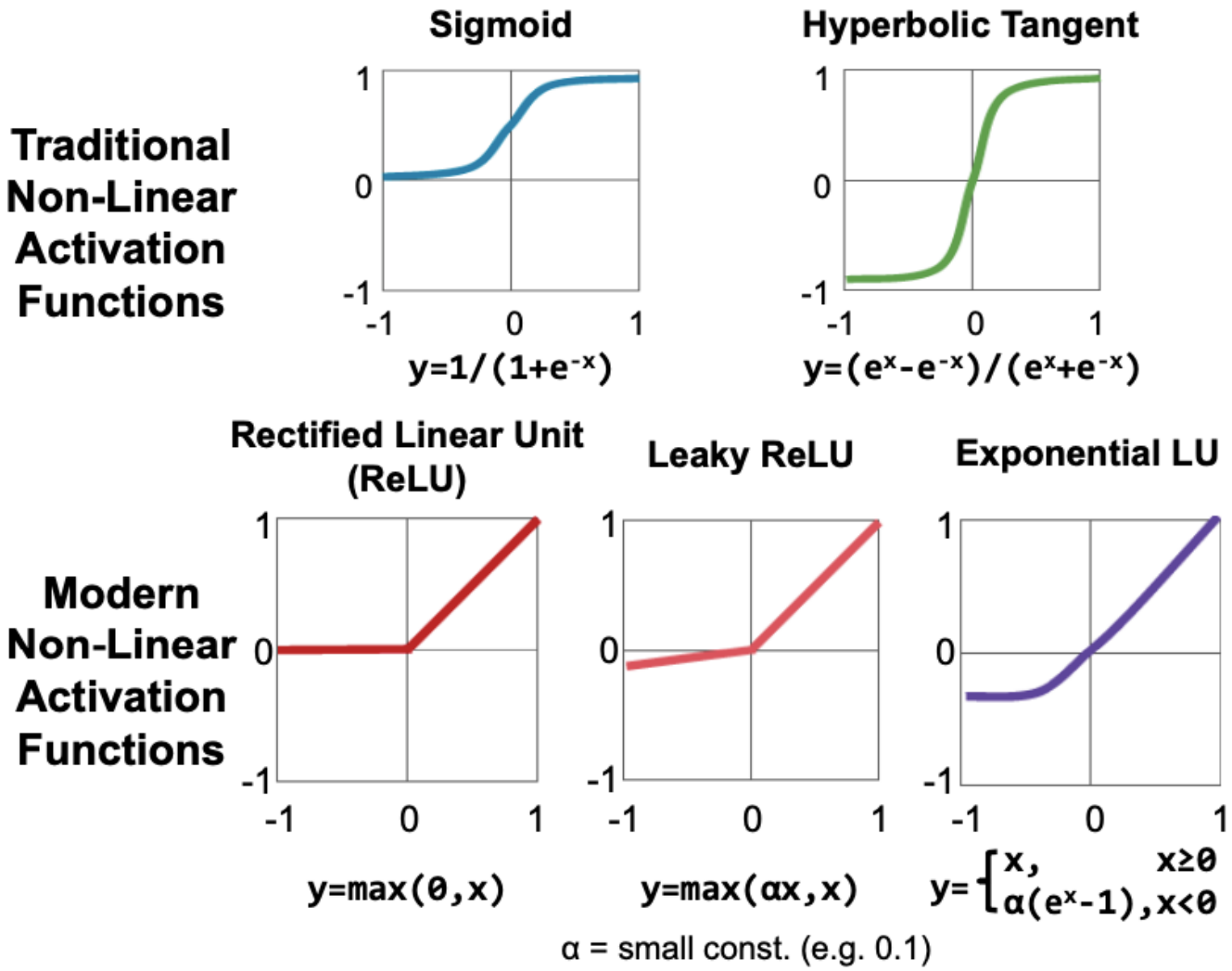


Deep Neural Networks

- Dataset
  - Problem dependent
  - https://github.com/pytorch/vision
  - Training set, validation set, test set
- Cost function
  - Similar to other parametric ML models
  - Backpropagation (using chain rule)
- Optimization
  - Stochastic Gradient Descent

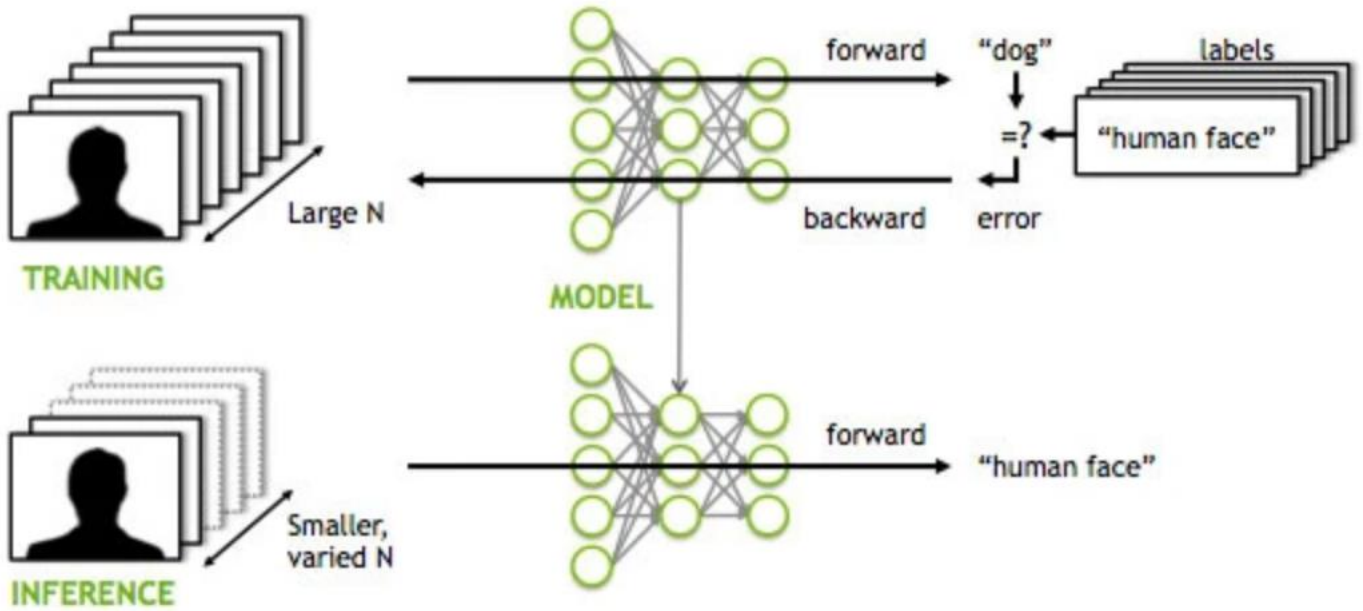
DNN Non-Linearity

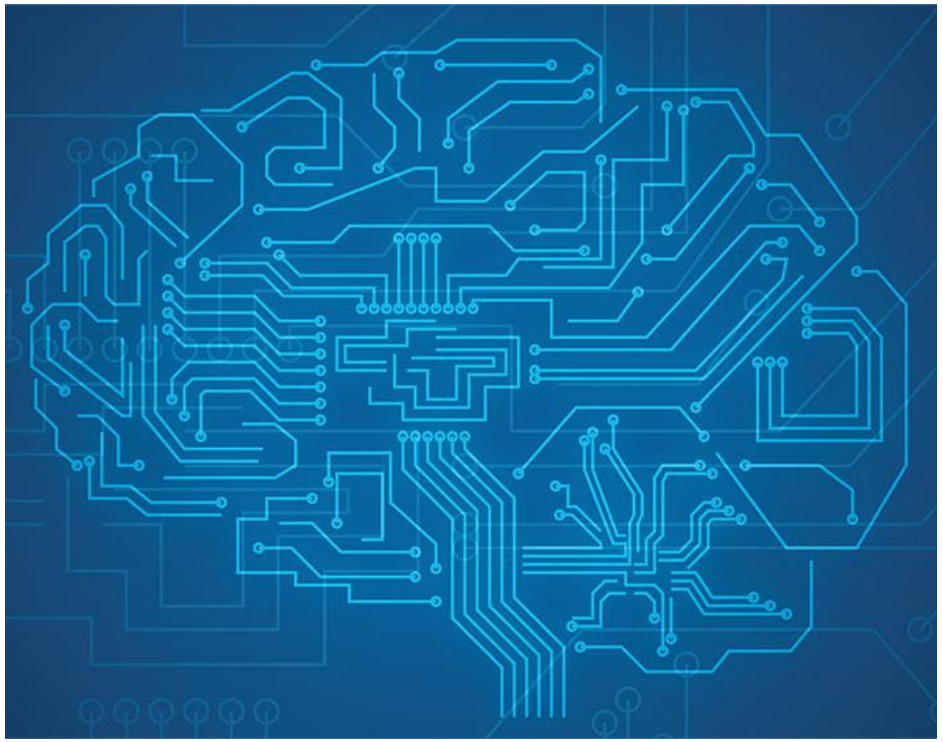
- Rectified Linear Units (ReLU)
  - $y(x) = \max\{0, x\}$
- Benefits:
  - Reduce the likelihood of the gradient vanishing problem.
  - Adding more sparsity/regularization
  - Easy to compute



Training vs Inference

- Training
  - Dataset
  - Cost function
  - Optimization function
  - Model
- Inference
  - Dataset
  - Model





DNNs

- AI, ML, and DL
- ML Basics
- DL Overview
- **AlexNet Example**

AlexNet Model

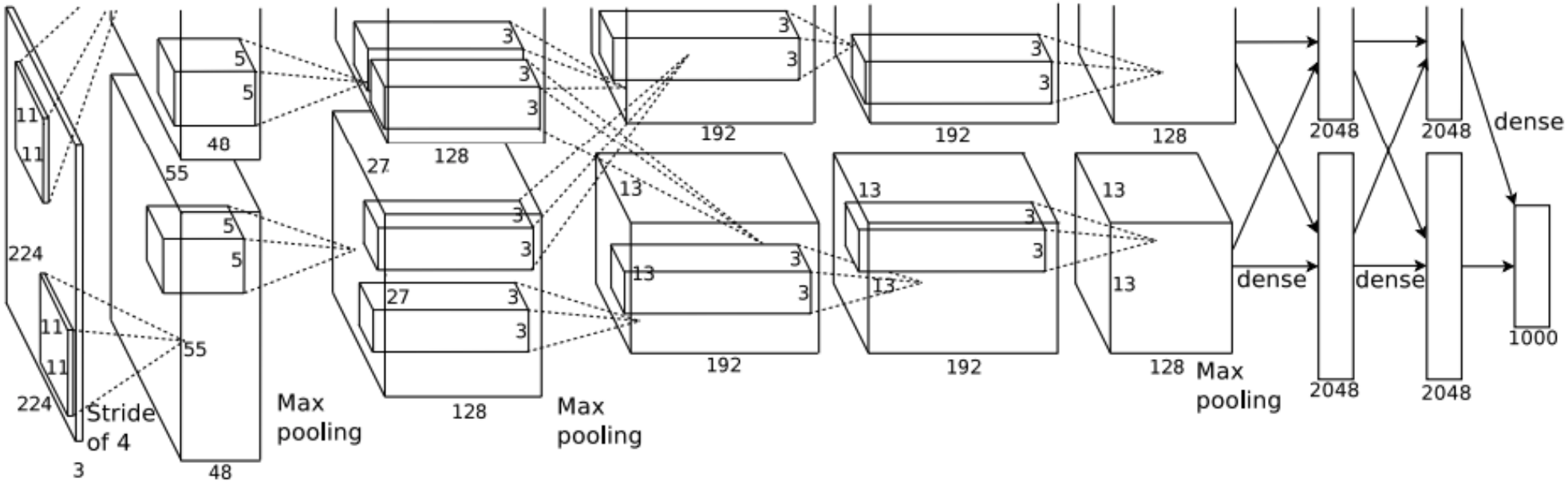
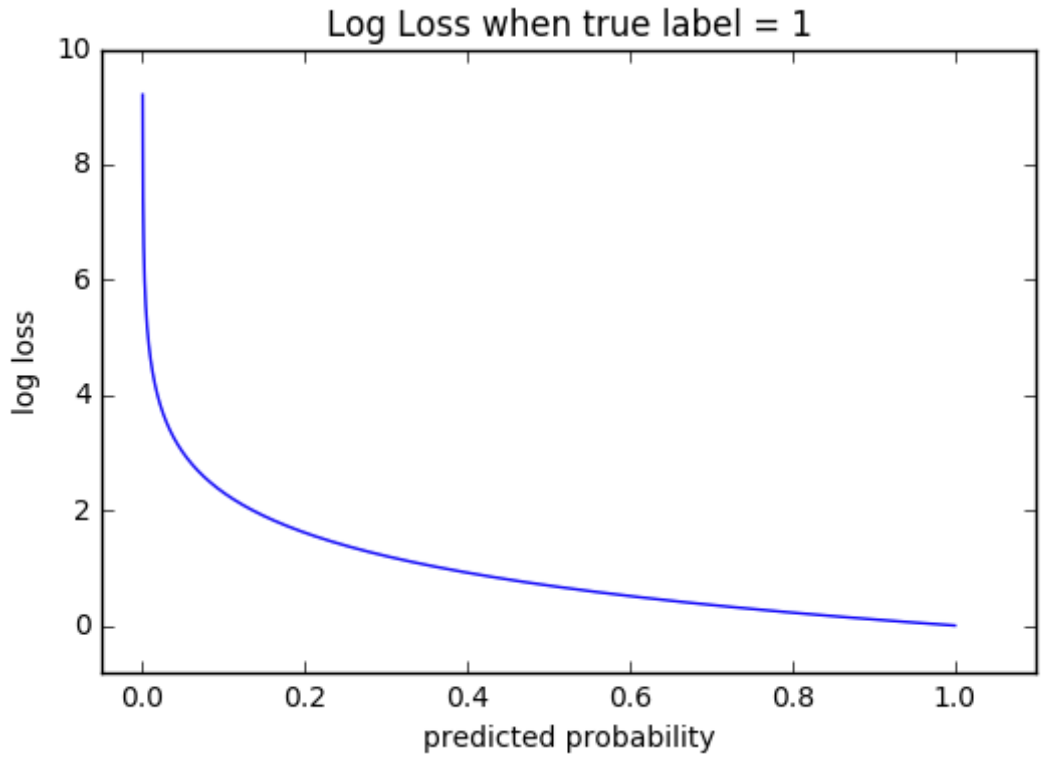


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

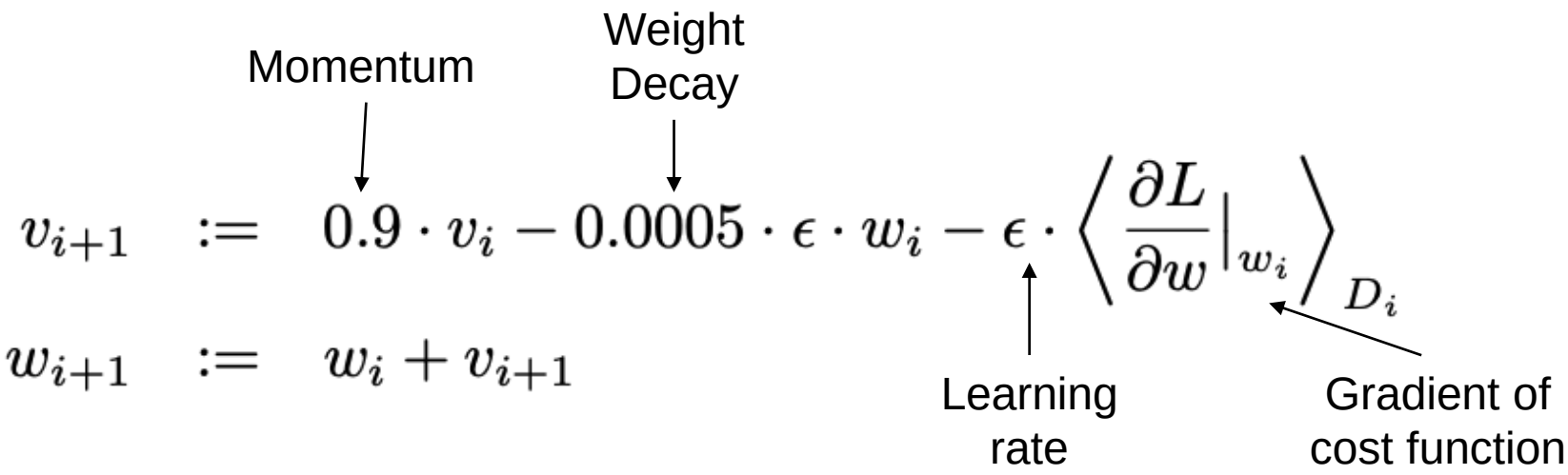
AlexNet Cost (loss) function

- Minimize the cross-entropy loss function
- Measures the performance of a classification model whose output is a probability value between 0 and 1
- $CF = -\frac{1}{N}\left(\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i)\right)$



AlexNet Optimization Method

- Stochastic Gradient Descent
  - Batch size = 128
  - Update rule:



State-of-the-art Training Time

	Batch Size	Processor	DL Library	Time	Accuracy
He et al.	256	Tesla P100 x8	Caffe	29 hours	75.30%
Goyal et al.	8K	Tesla P100 x256	Caffe2	1 hour	76.30%
Smith et al.	8K→16K	full TPU Pod	TensorFlow	30 mins	76.10%
Akiba et al.	32K	Tesla P100 x1024	Chainer	15 mins	74.90%
Jia et al.	64K	Tesla P40 x2048	TensorFlow	6.6 mins	75.80%
This work	**34K→68K**	**Tesla V100 x2176**	**NNL**	**224 secs**	**75.03%**

https://news.developer.nvidia.com/sony-breaks-resnet-50-training-record-with-nvidia-v100-tensor-core-gpus/