

15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 13:
Convolutional Neural Nets

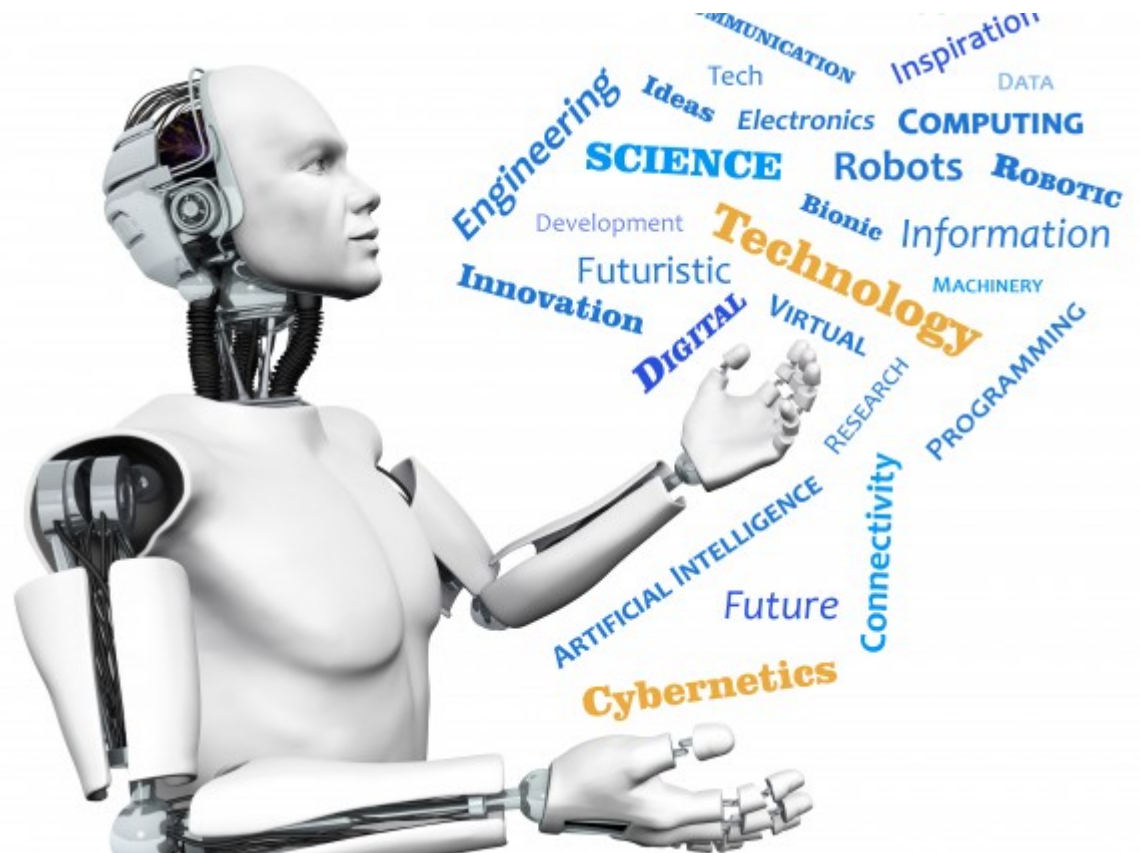
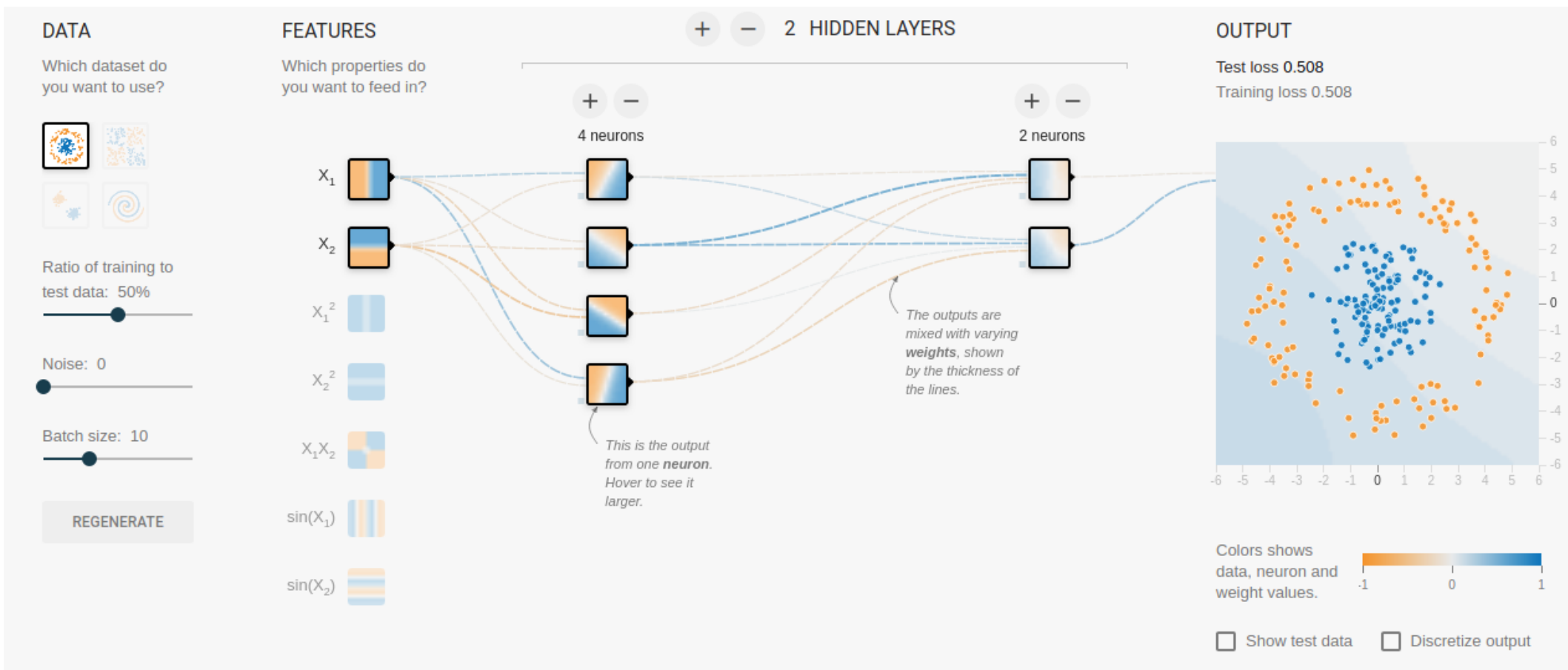


Image from <http://www.futuristgerd.com/2015/09/10>

TensorFlow Playground

Google's interactive backprop simulator.
<https://playground.tensorflow.org>

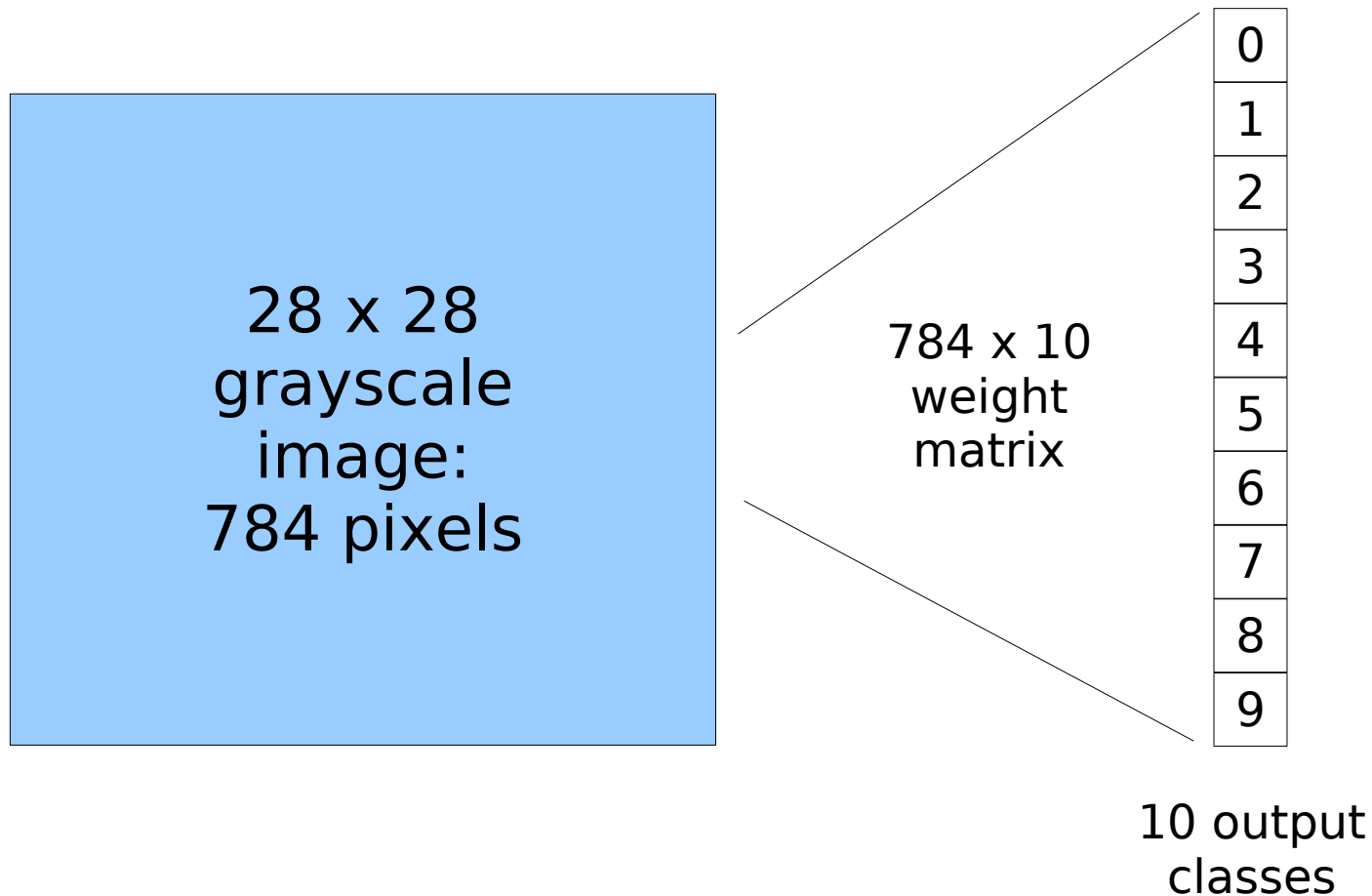


MNIST Dataset

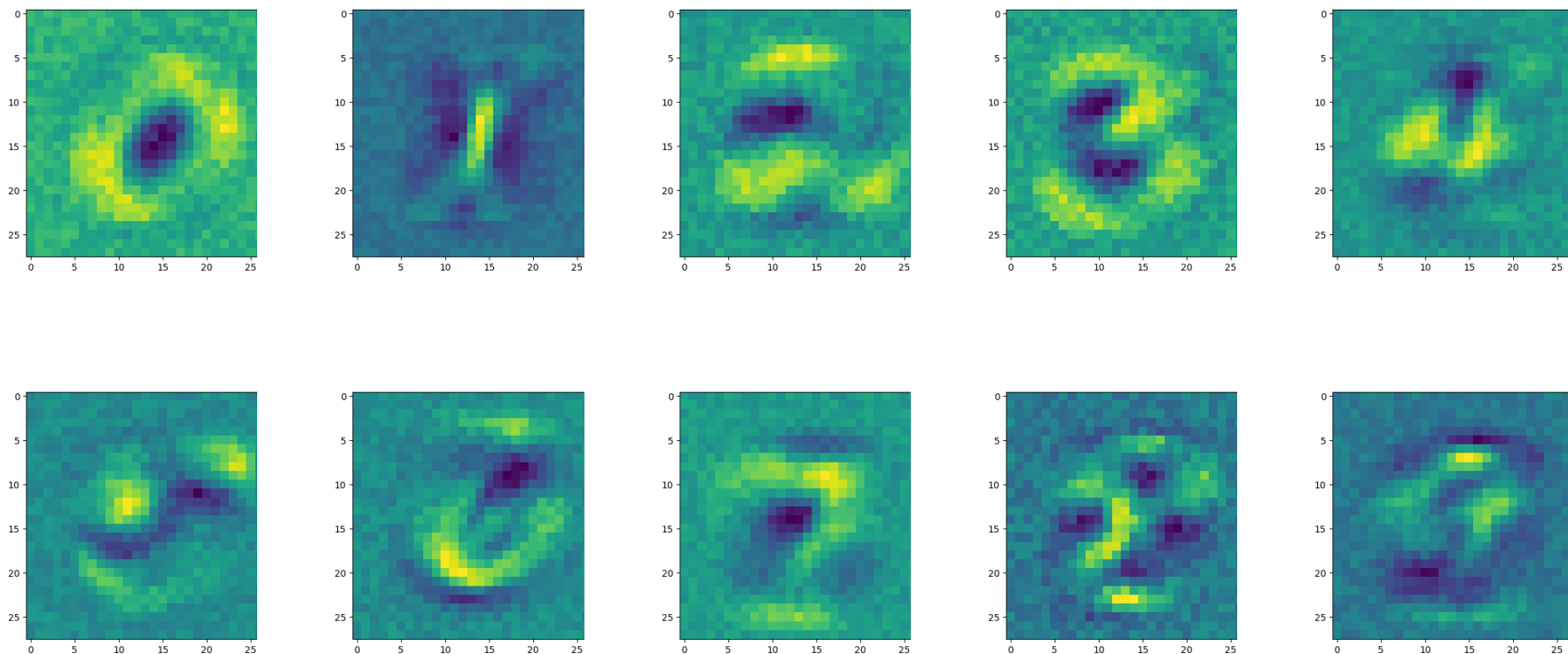
- 60,000 labeled handwritten digits
- 28 x 28 pixel grayscale images



Recognition With a Linear Network



Learned Weights to Output Units

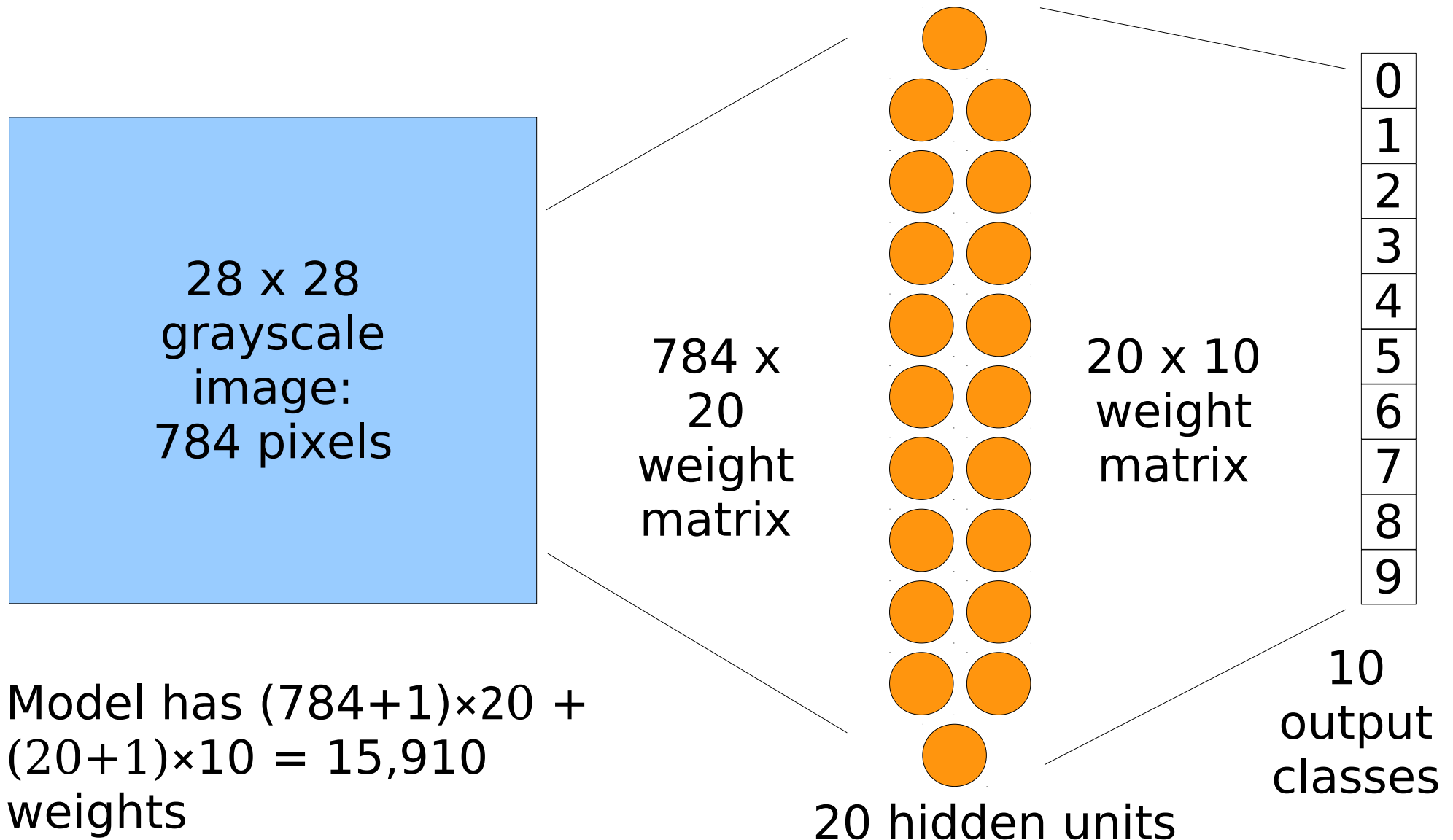


Training set performance: 89% correct.

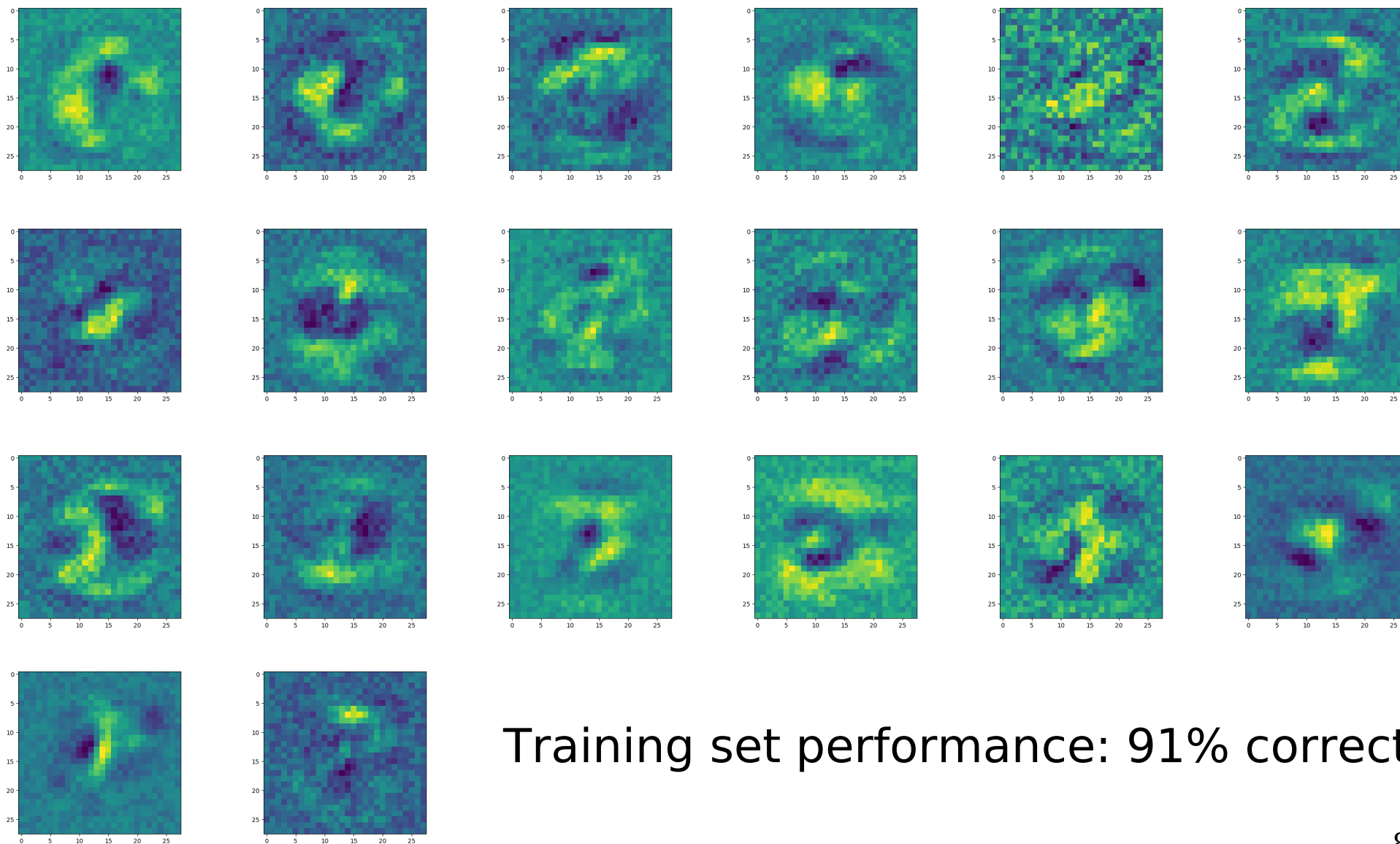
Batch Size

- An *epoch* is one pass through all the training data.
- With a large training set (60,000 images), we don't need to see all the training examples in order to estimate the error gradient.
- We set a batch size of 100 to indicate we want to do a weight update after every 100 training examples.
 - The examples need to be mixed together.
 - What if we trained on all the 2's first?

Adding A Hidden Layer

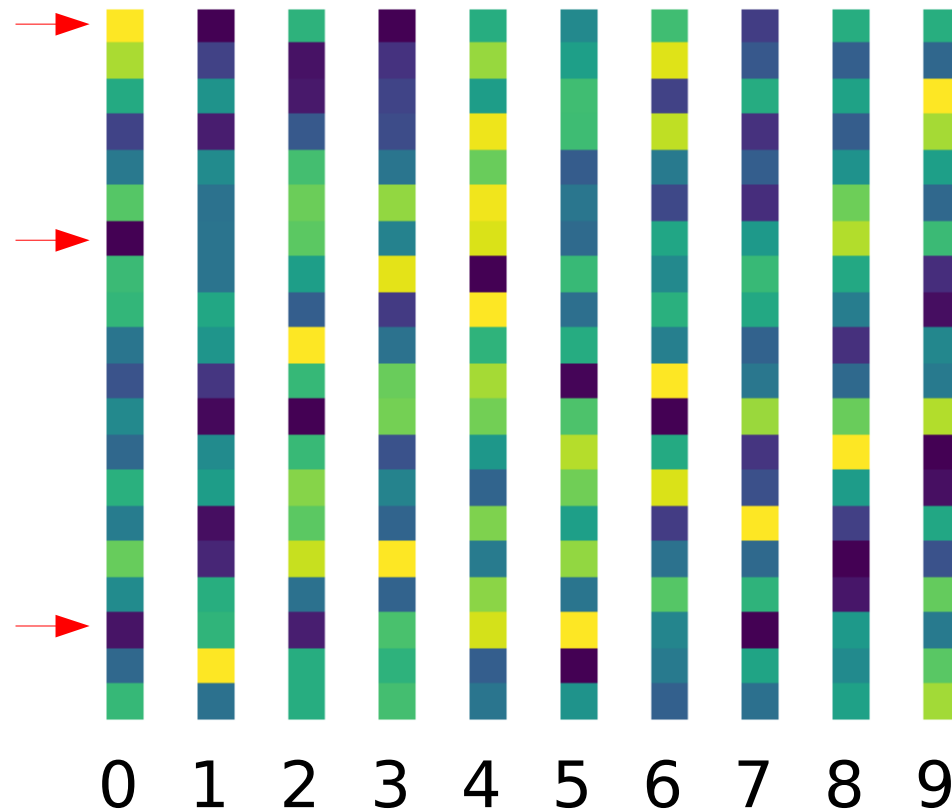


Learned Weights to Hidden Units



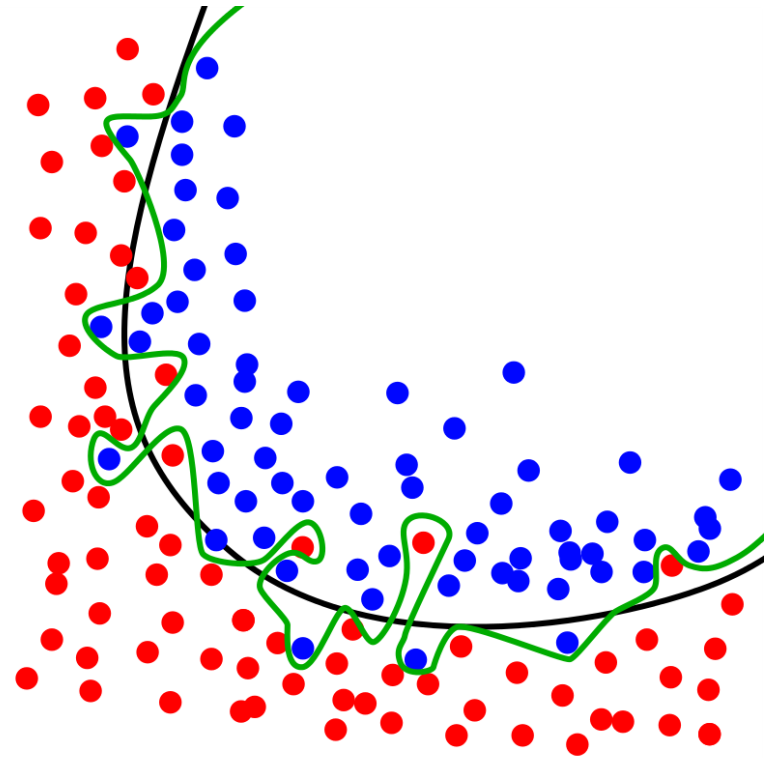
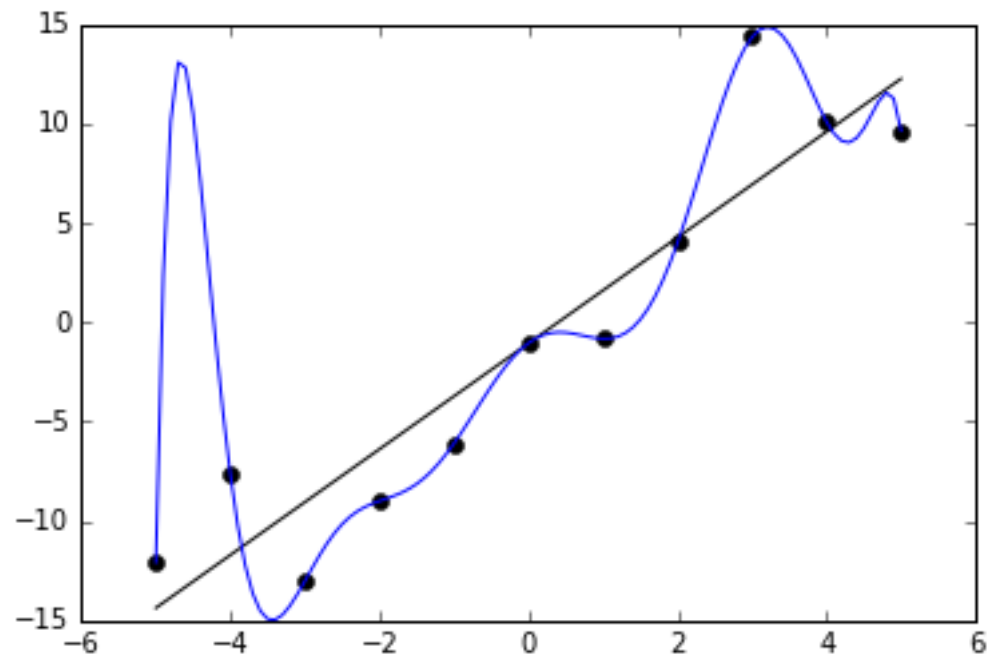
Training set performance: 91% correct.

Learned Weights to Output Units



Training set performance: 91% correct.

Overfitting

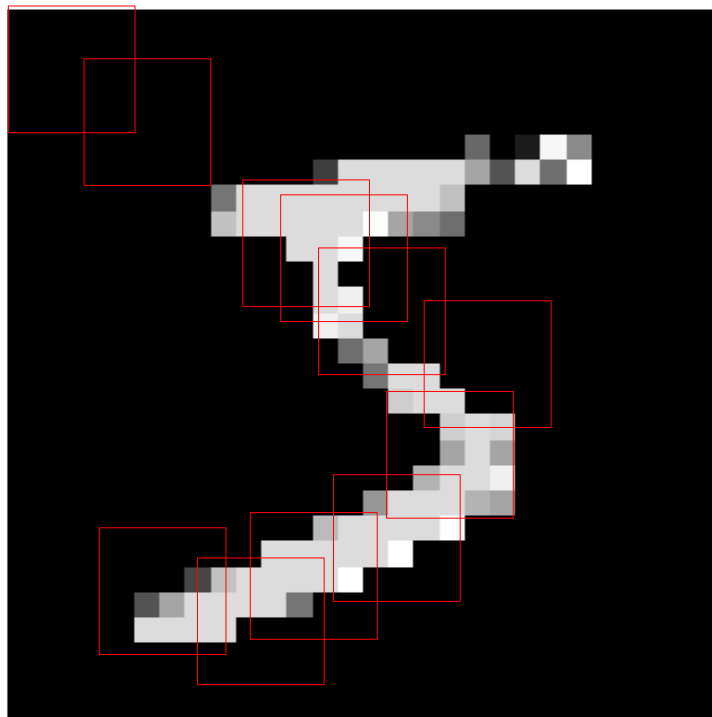


How to Avoid Overfitting

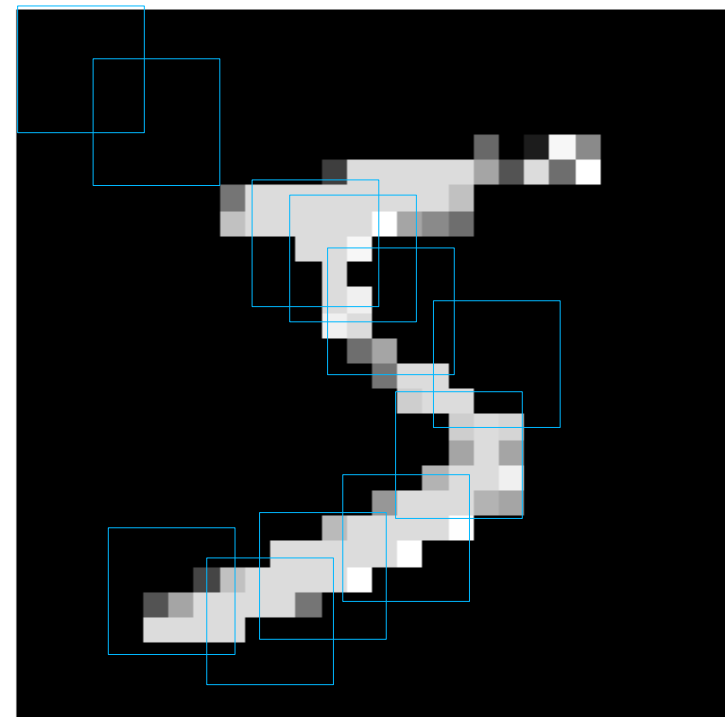
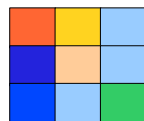
- Increase the size of the training set.
- Reduce the number of parameters:
 - Fewer hidden units
 - Shared weights (convolutional network)
- Regularization: penalize large weights to encourage making more weights be zero.
- Dropout: randomly disable some fraction of the connections on every iteration.
- Early stopping:
 - Maintain a separate cross-validation set
 - Stop training when the cv error rises

Convolutional Neural Networks

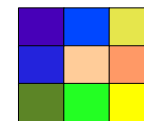
- Learn small (3x3 or 5x5) feature detectors or *kernels* that can be applied anywhere in the image.



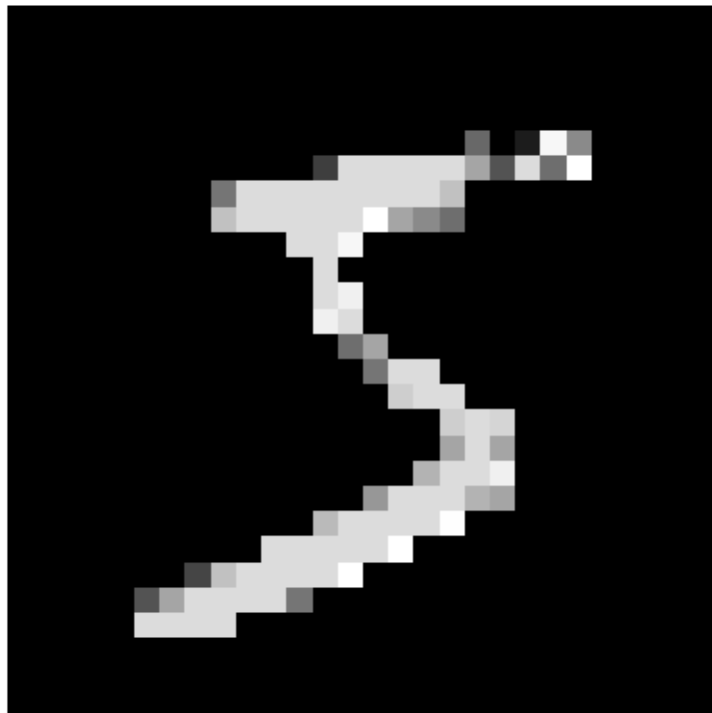
Feature 1:



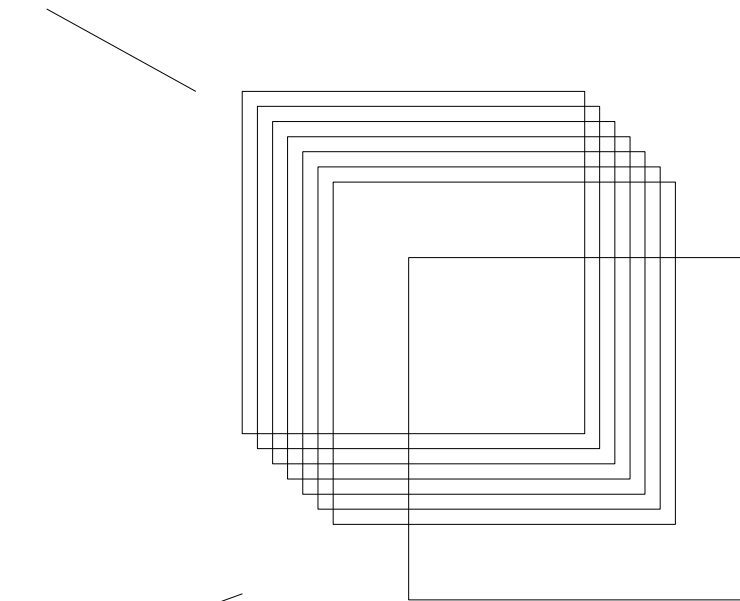
Feature 2:



Feature Maps



28 x 28 image



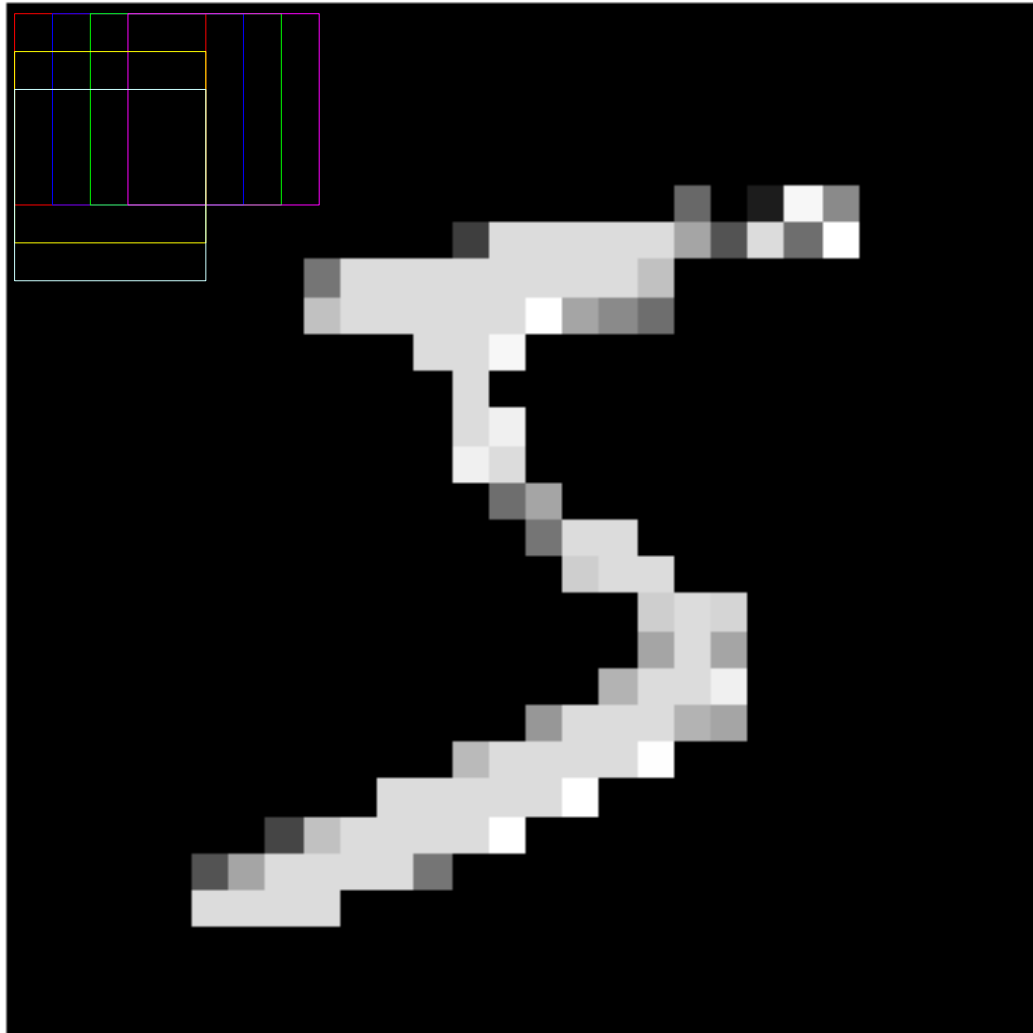
32 feature maps
26 x 26

32 kernels
5x5 pixels
stride 1
padding 1

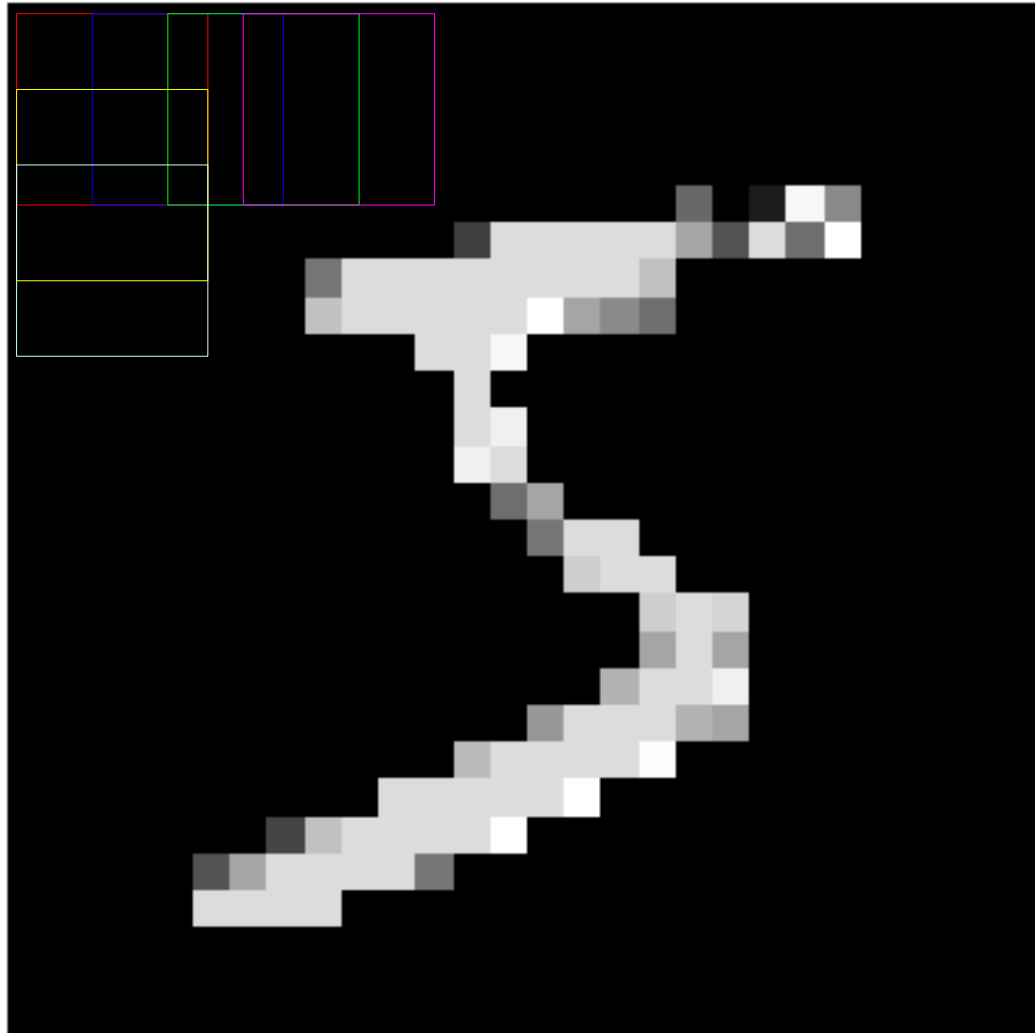
weights = $32 \times (5 \times 5 + 1) = 832$ (small!)

connections = $32 \times (26 \times 26) \times (5 \times 5 + 1) = 562,432$

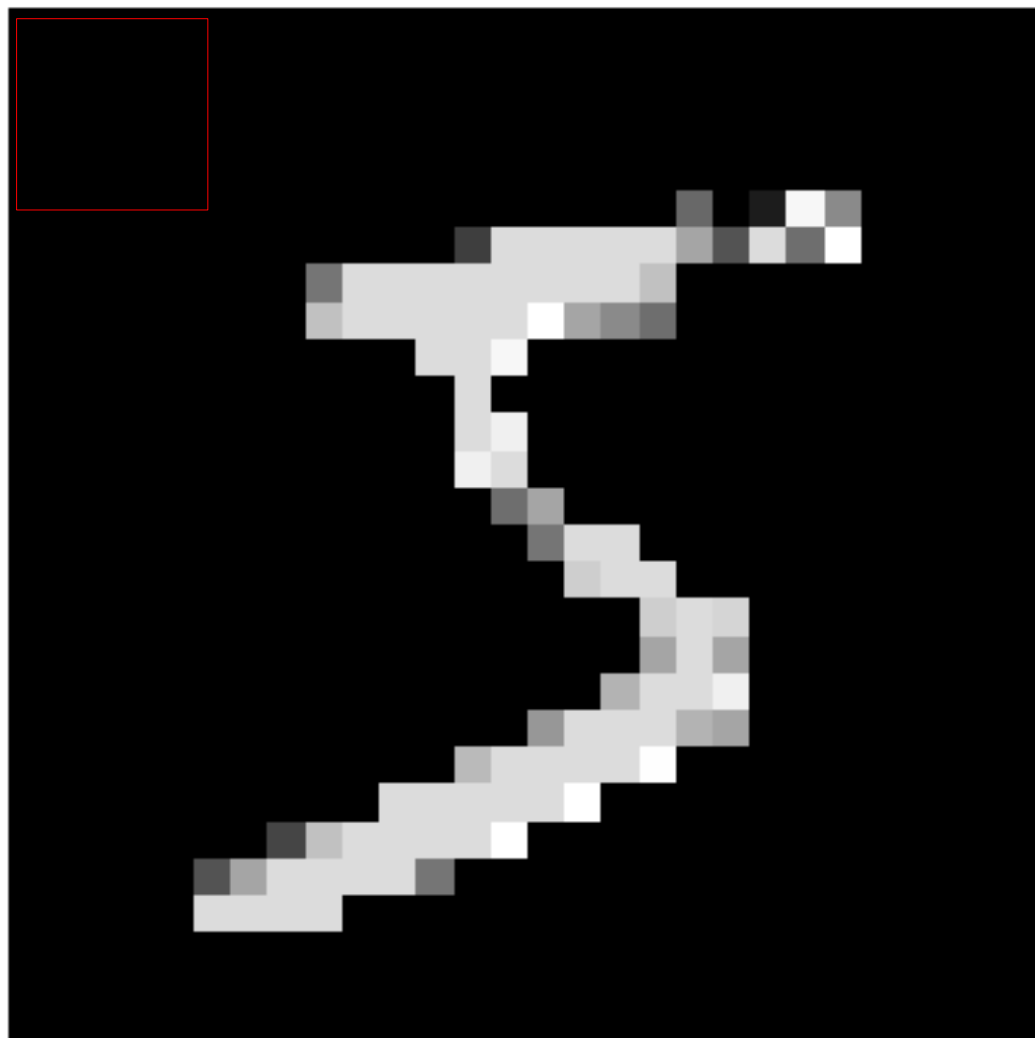
Stride 1



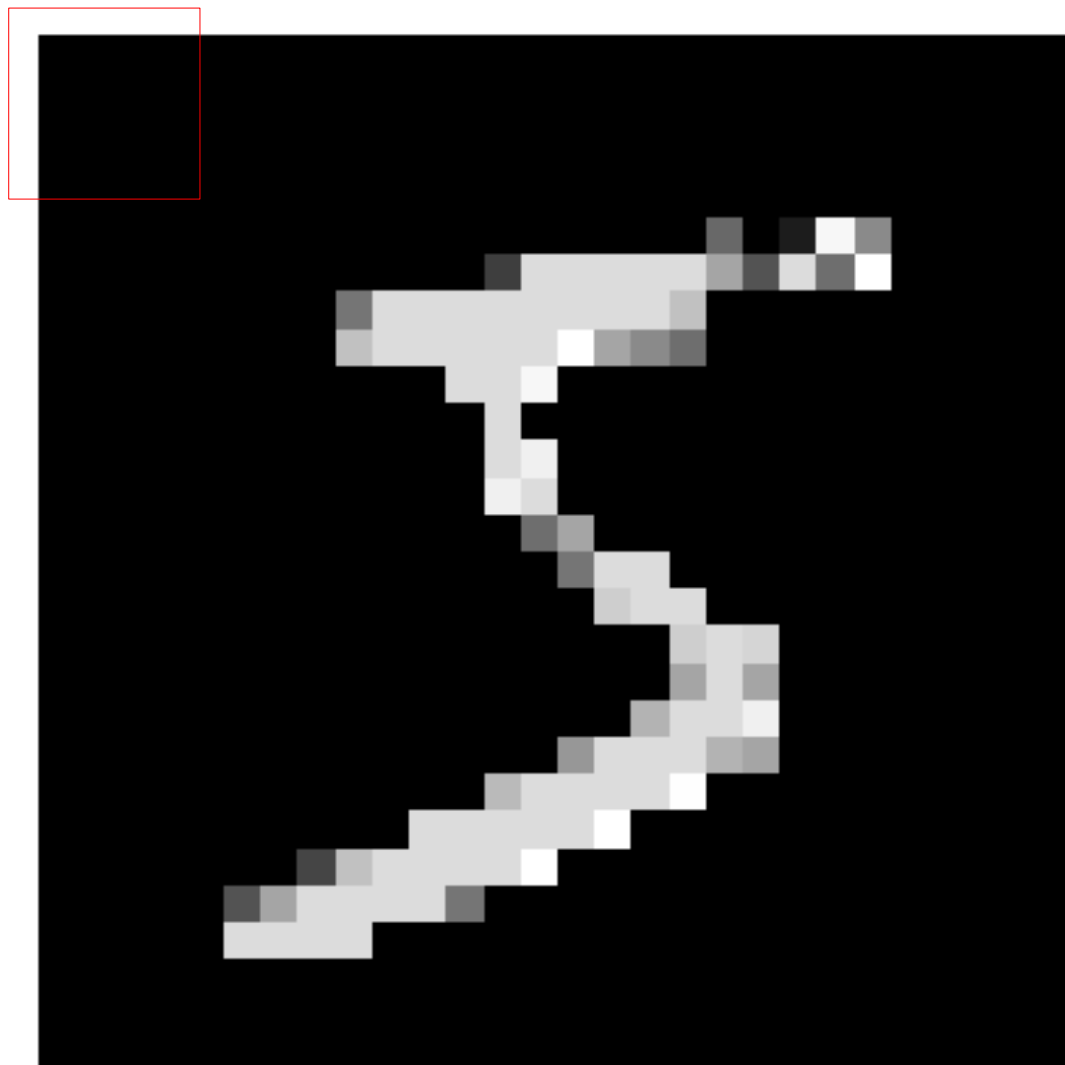
Stride 2



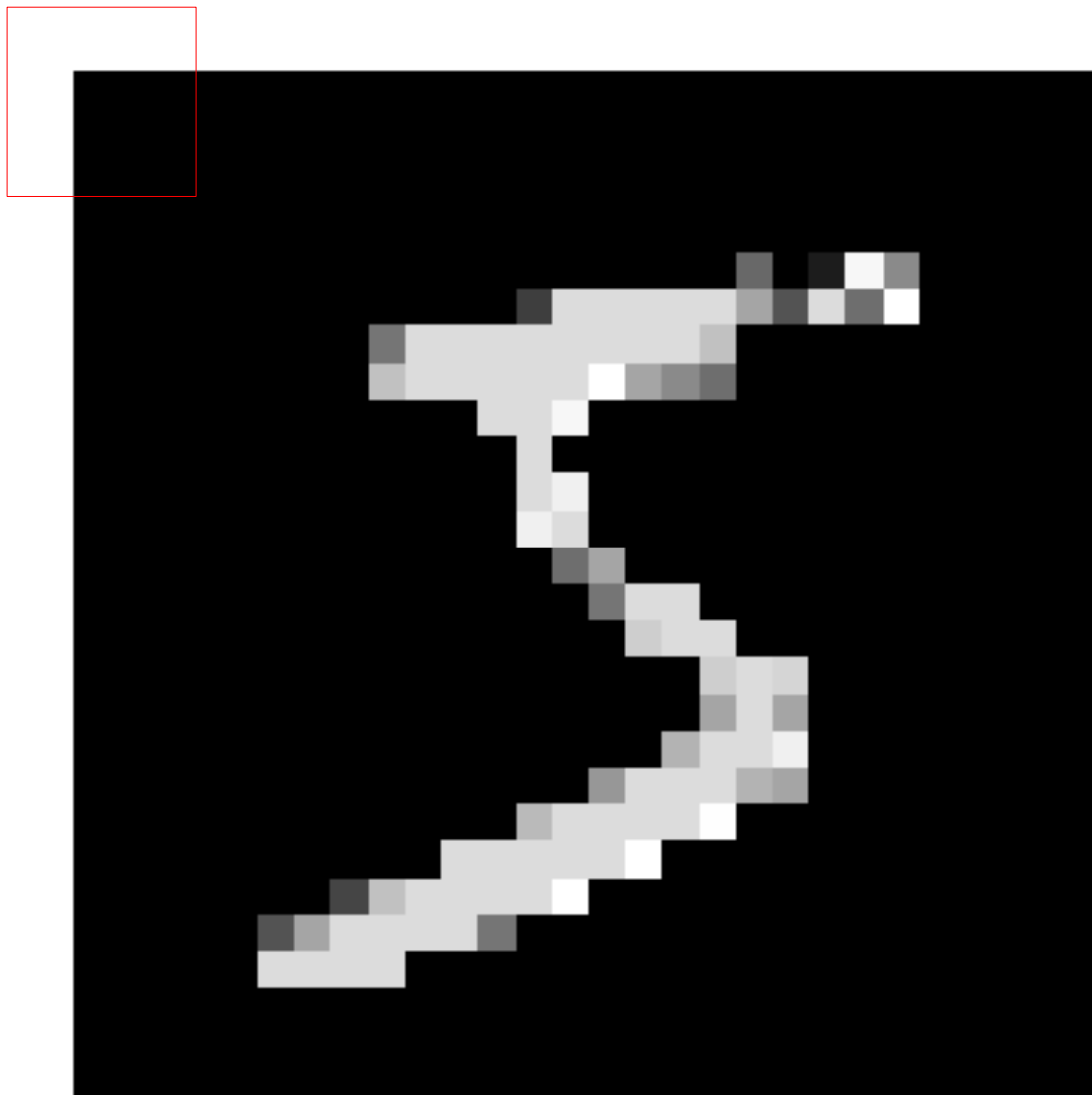
First Kernel: Padding 0



First Kernel: Padding 1

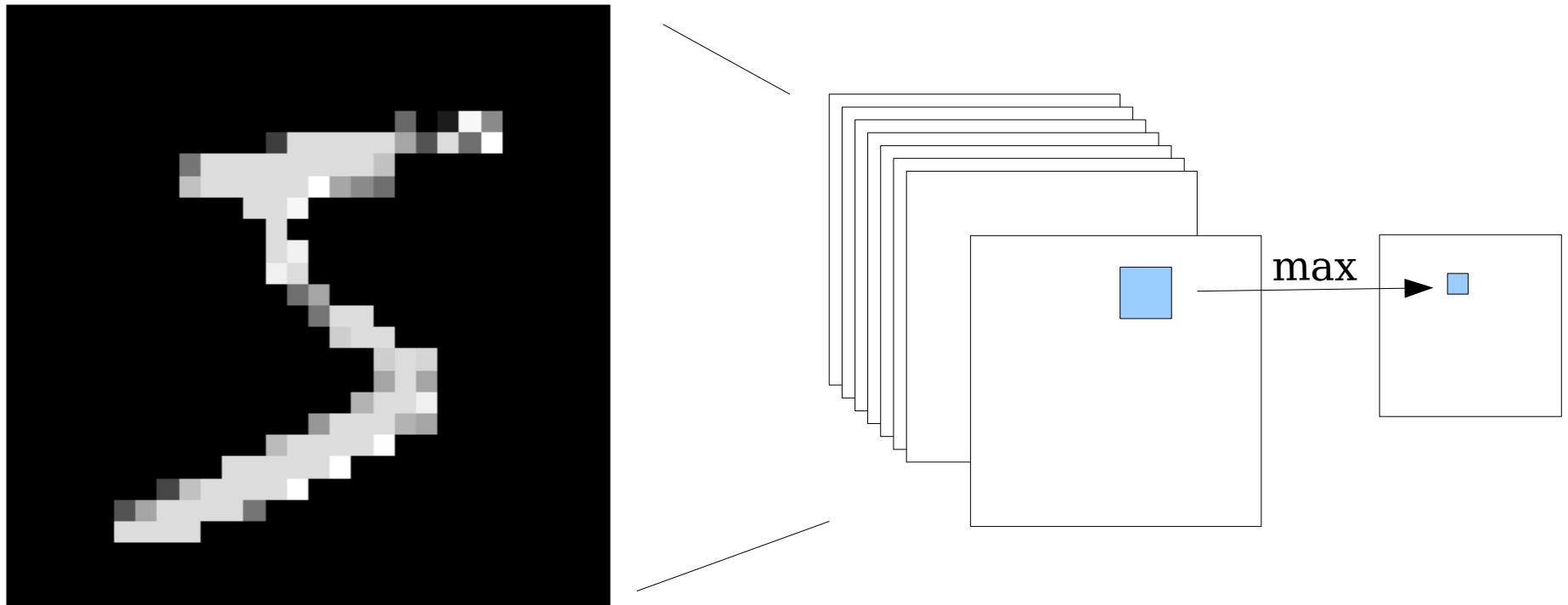


First Kernel: Padding 2



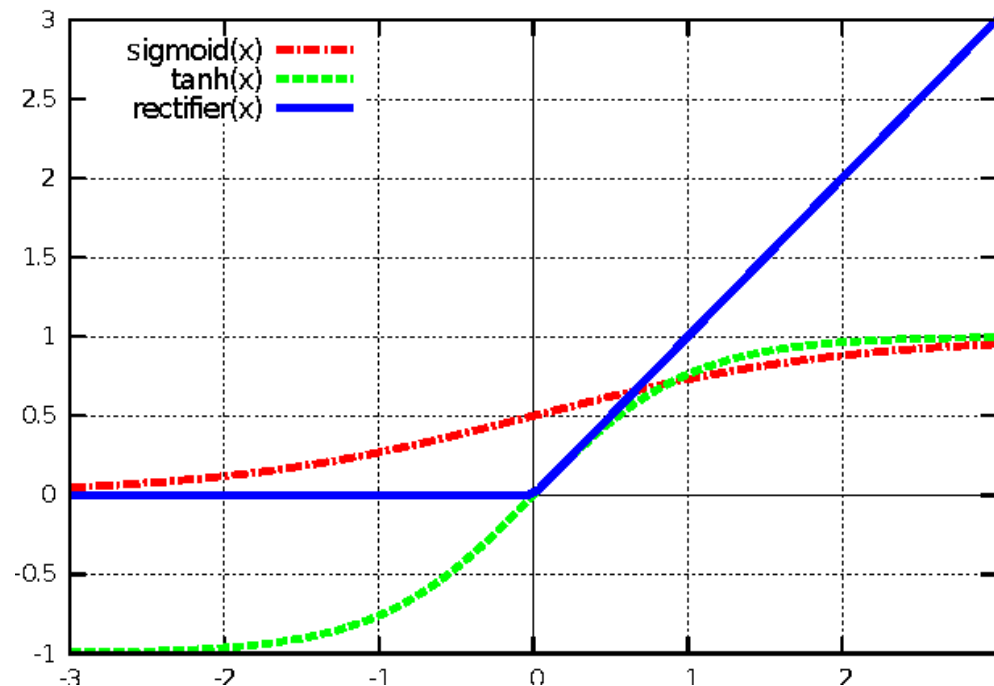
Max Pooling

- We might not care exactly where a feature appears in the image.
- Downsampling by max pooling reduces the number of units and connections.



Choice of Activation Function

- Sigmoid and tanh were popular early on:



- Now it's more common to use ReLU:
Rectified Linear Unit. $g(x) = \max(x, 0)$
 - Derivative doesn't go to zero for large x.

Choice of Loss Function

- Mean Squared Error is a general loss function but not always the best to use.

$$E = \frac{1}{2} \sum_p (d^p - y^p)^2$$

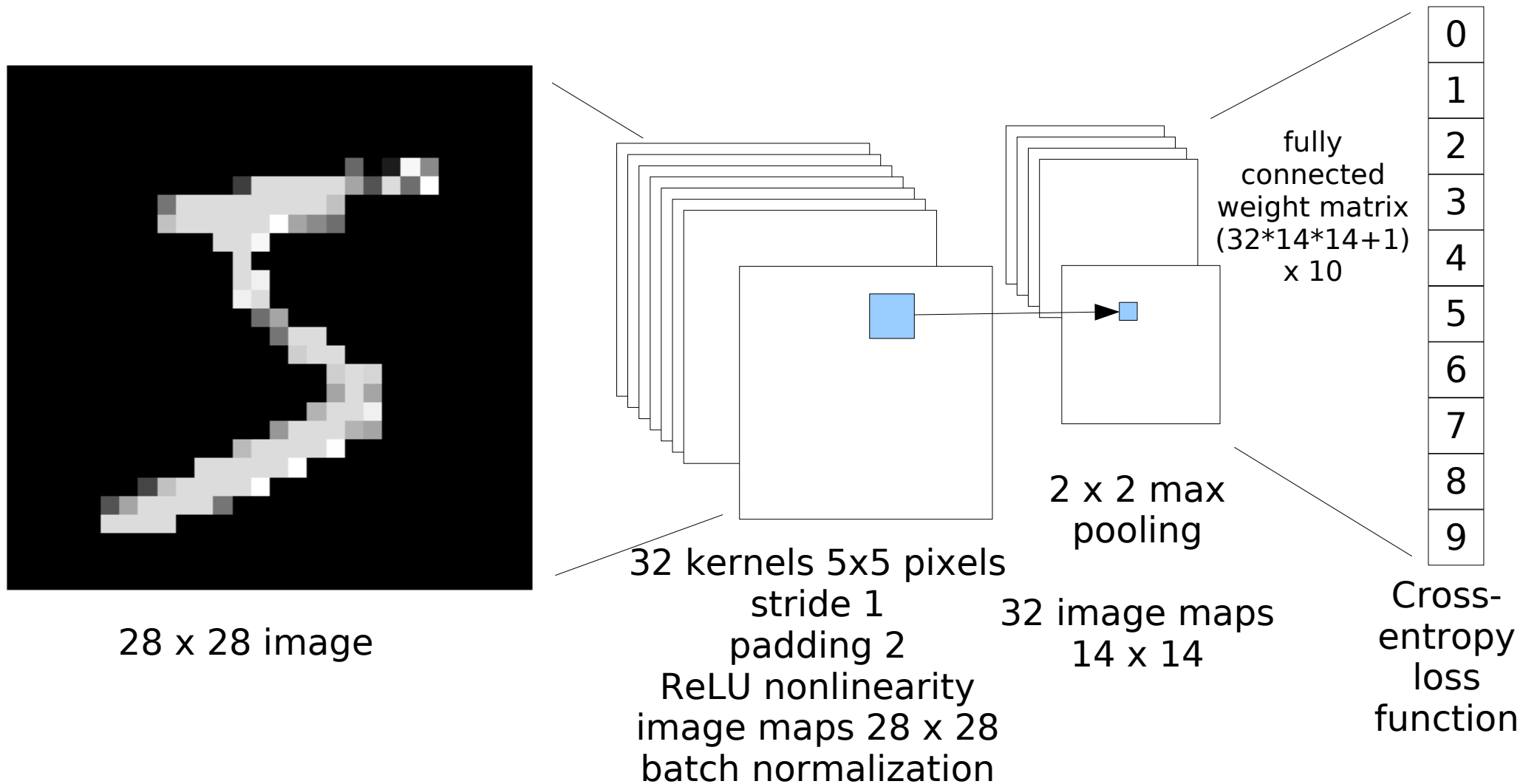
- If desired outputs are probabilities (values between 0 and 1), use cross-entropy instead. Heavily penalizes really wrong outputs.

$$E = \sum_p -d^p \log(y^p) - (1 - d^p) \log(1 - y^p)$$

Batch Normalization

- We want the activity patterns in each layer to have nice statistical properties (mean and variance) because this helps speed up learning.
- But each weight update changes the statistical distribution.
- Solution: “batch normalization”, a trick for making the distributions more uniform.
- Built in to PyTorch.

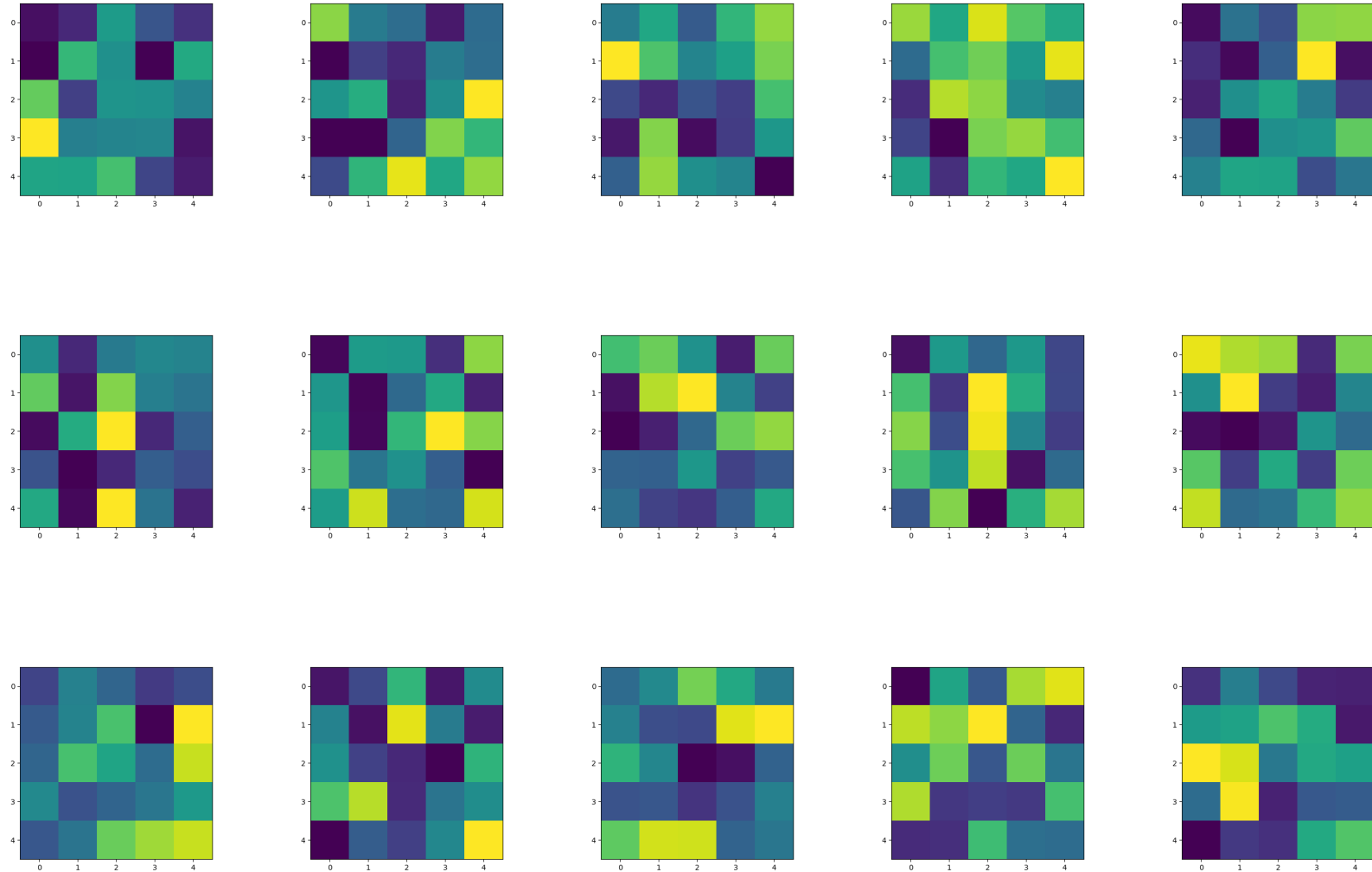
MNIST With A CNN



parameters = 63,626
How many connections?

Accuracy on training set: 98.3%

Sample Learned Kernels

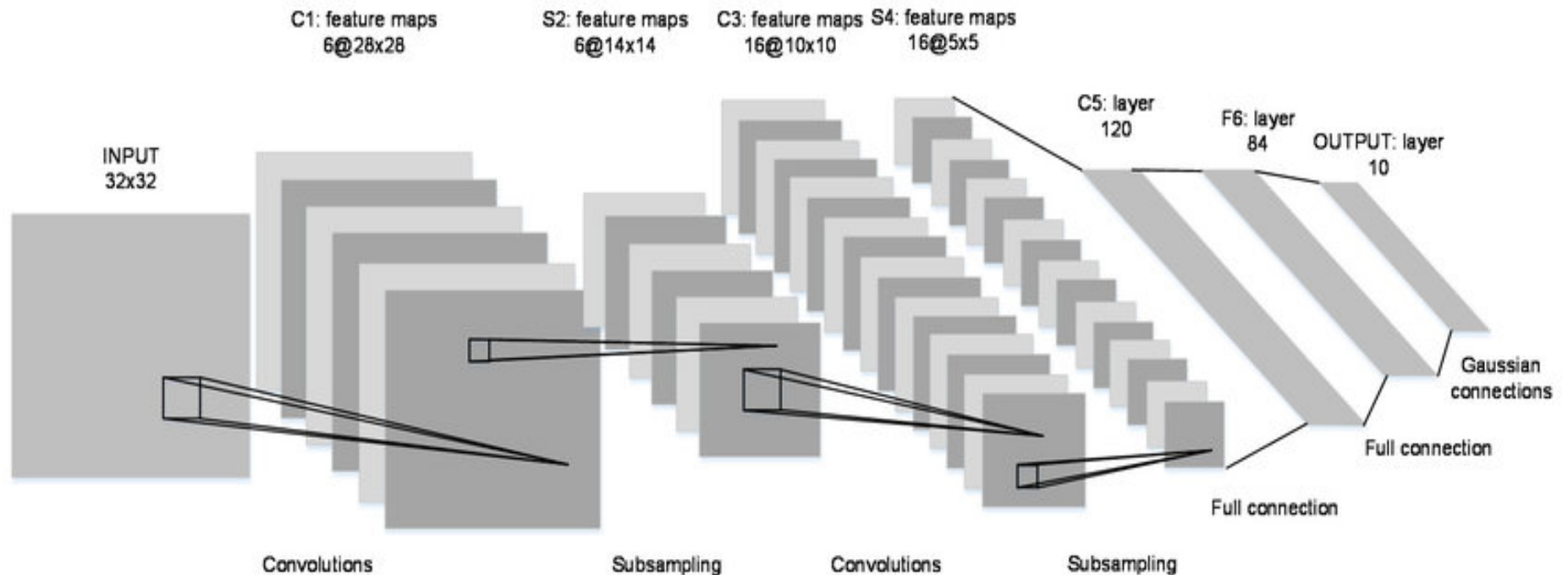


Deep Neural Networks

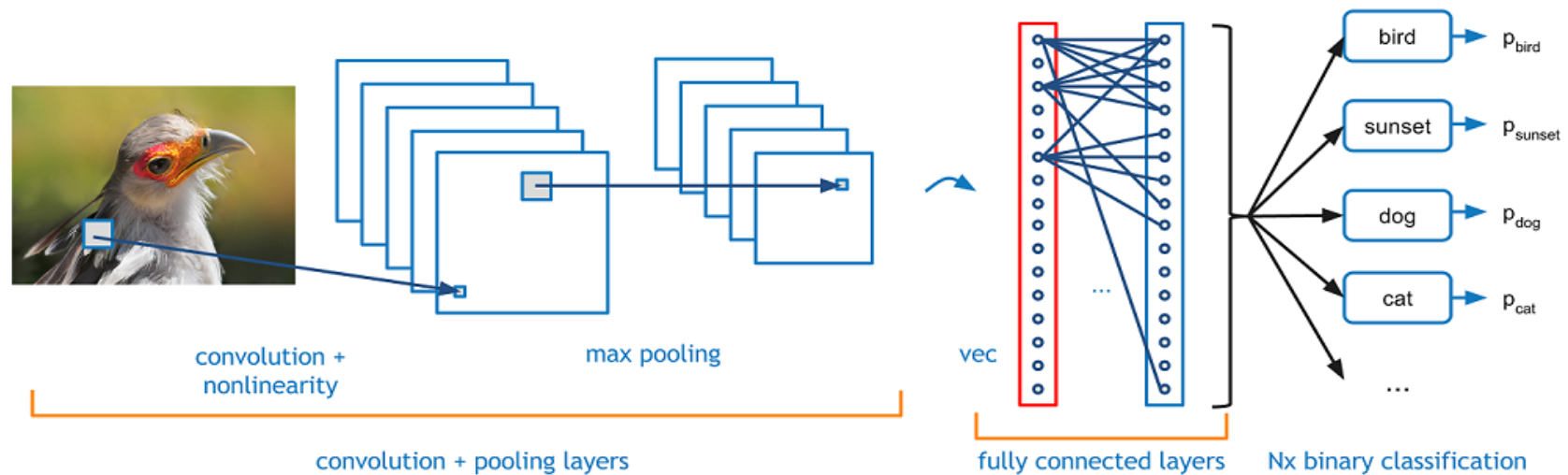
- For really hard problems (e.g., object recognition on color images) we may need many layers.
- Series of convolutional and max pooling layers, followed by some fully connected layers.
 - LeNet had 10 layers.
 - Inception V1 had 27 layers.
 - ResNet has 100 layers.
- GPUs required for training.

LeNet (Yann LuCun, 1990s)

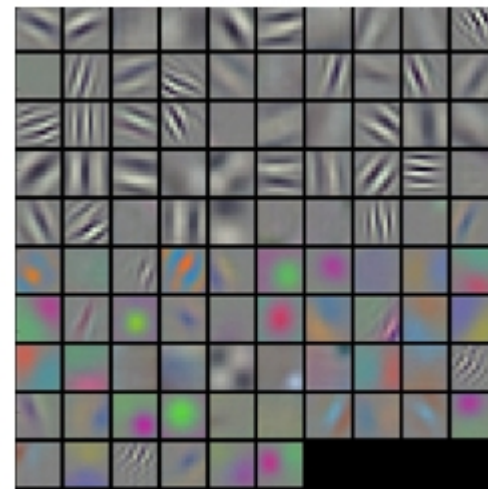
- Handwritten digit recognition



Object Recognition CNN



<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>



Visualizations of filters

PyTorch

- Python package for tensor manipulation and vectorized computations, including neural net learning.
 - Replacement for numpy
 - Optimized for GPUs
- Tensors are multi-dimensional arrays, similar to numpy's ndarray structure.
- Code can run on either CPU or GPU.