CSC411/2515 Fall 2016

Neural Networks Tutorial

Lluís Castrejón

Oct. 2016

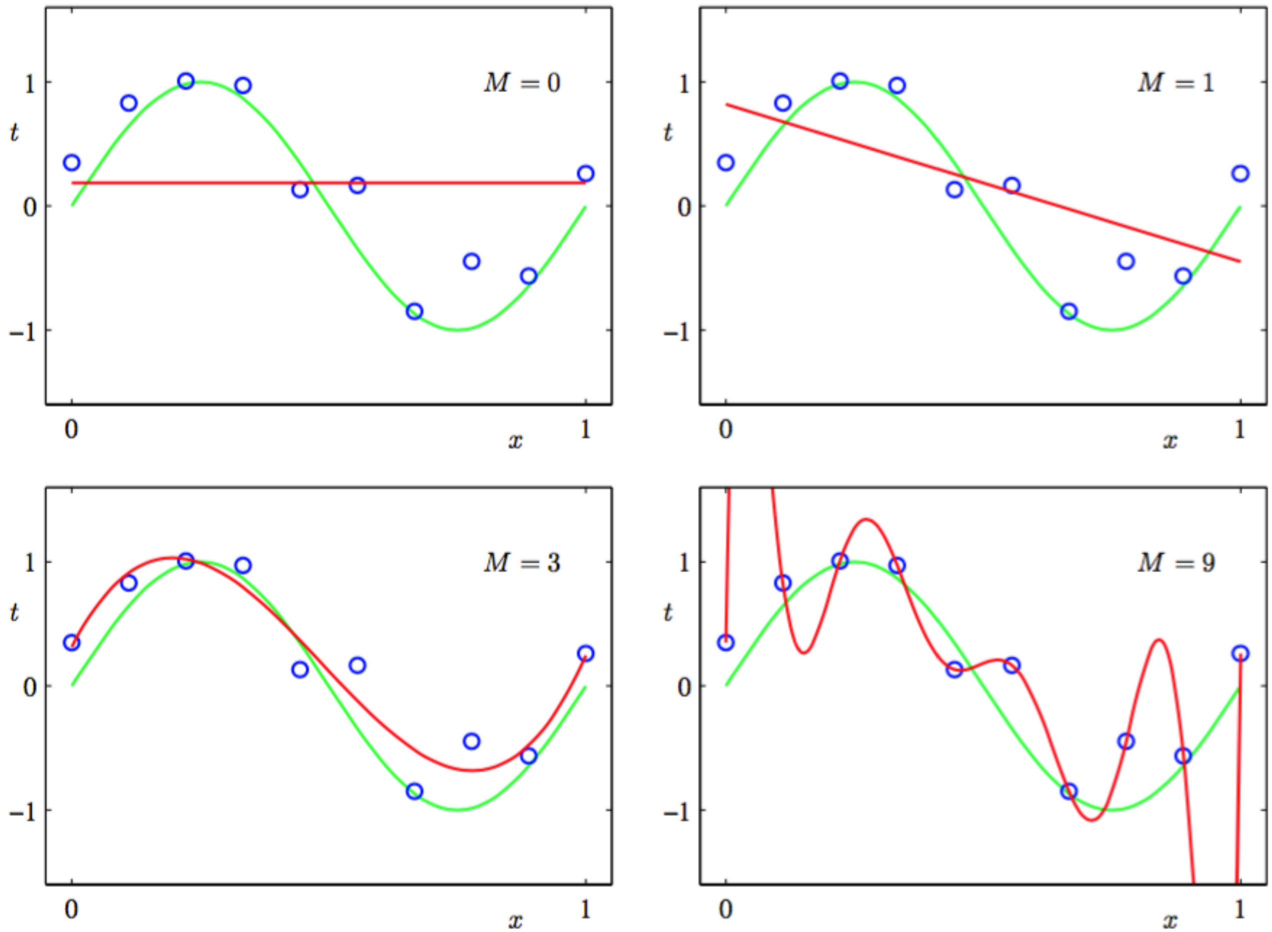
Slides adapted from Yujia Li's tutorial and Prof. Zemel's lecture notes.

Overfitting

- The training data contains information about the regularities in the mapping from input to output. But it also contains noise
  - The target values may be unreliable.
  - There is sampling error. There will be accidental regularities just because of the particular training cases that were chosen

- When we fit the model, it cannot tell which regularities are real and which are caused by sampling error.
  - So it fits both kinds of regularity.
  - If the model is very flexible it can model the sampling error really well. This is a disaster.

Overfitting



Picture credit: Chris Bishop. Pattern Recognition and Machine Learning. Ch.1.1.

Preventing overfitting

- Use a model that has the right capacity:
  - enough to model the true regularities
  - not enough to also model the spurious regularities (assuming they are weaker)

- Standard ways to limit the capacity of a neural net:
  - Limit the number of hidden units.
  - Limit the size of the weights.
  - Stop the learning before it has time to overfit.

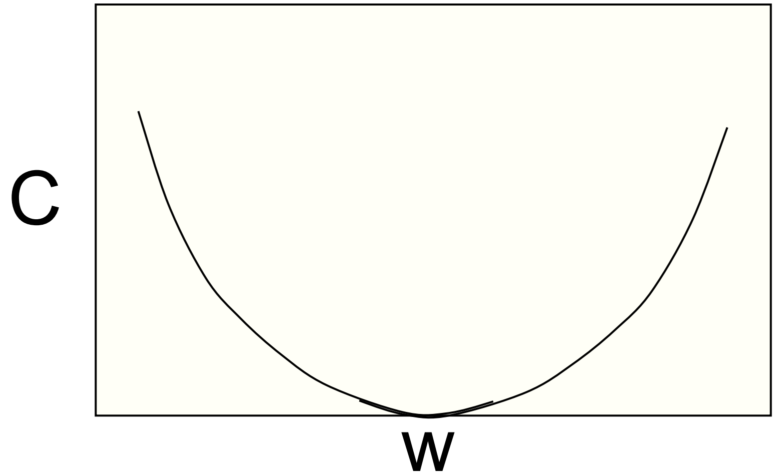
Limiting the size of the weights

Weight-decay involves adding an extra term to the cost function that penalizes the squared weights.

- Keeps weights small unless they have big error derivatives.

$$C = E + \frac{\lambda}{2} \sum_i w_i^2$$

$$\frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$$

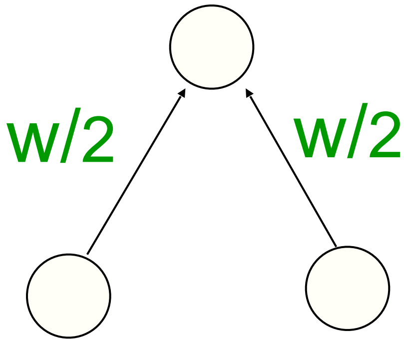


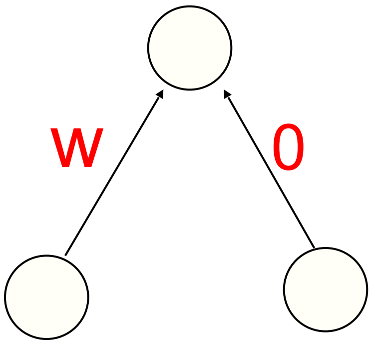
$$when \quad \frac{\partial C}{\partial w_i} = 0, \quad w_i = -\frac{1}{\lambda} \frac{\partial E}{\partial w_i}$$

The effect of weight-decay

- It prevents the network from using weights that it does not need
  - This can often improve generalization a lot.
  - It helps to stop it from fitting the sampling error.
  - It makes a smoother model in which the output changes more slowly as the input changes.

- But, if the network has two very similar inputs it prefers to put half the weight on each rather than all the weight on one → other form of weight decay?





Deciding how much to restrict the capacity

- How do we decide which limit to use and how strong to make the limit?
  - If we use the test data we get an unfair prediction of the error rate we would get on new test data.
  - Suppose we compared a set of models that gave random results, the best one on a particular dataset would do better than chance. But it won't do better than chance on another test set.

- So use a separate validation set to do model selection.

Using a validation set

- Divide the total dataset into three subsets:
  - Training data is used for learning the parameters of the model.
  - Validation data is not used of learning but is used for deciding what type of model and what amount of regularization works best
  - Test data is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data

- We could then re-divide the total dataset to get another unbiased estimate of the true error rate.

Preventing overfitting by early stopping

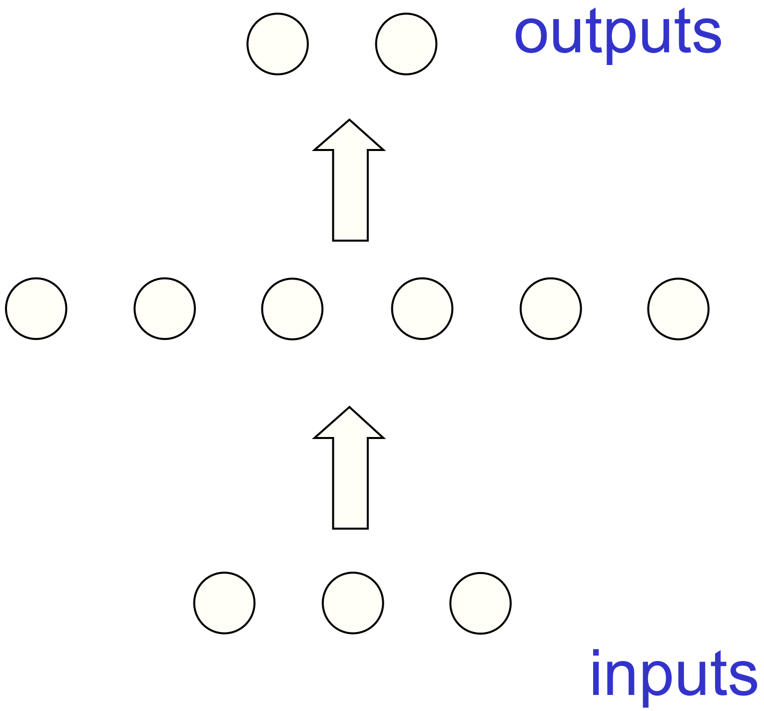
- If we have lots of data and a big model, its very expensive to keep re-training it with different amounts of weight decay

- It is much cheaper to start with very small weights and let them grow until the performance on the validation set starts getting worse

- The capacity of the model is limited because the weights have not had time to grow big.

Why early stopping works

- When the weights are very small, every hidden unit is in its linear range.
  - So a net with a large layer of hidden units is linear.
  - It has no more capacity than a linear net in which the inputs are directly connected to the outputs!
- As the weights grow, the hidden units start using their non-linear ranges so the capacity grows.



Le Net

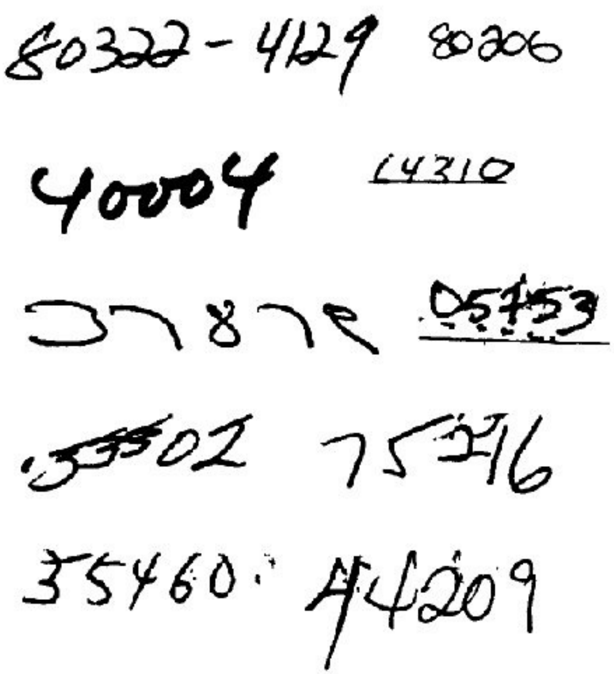
- Yann LeCun and others developed a really good recognizer for handwritten digits by using backpropagation in a feedforward net with:
  - Many hidden layers
  - Many pools of replicated units in each layer.
  - Averaging the outputs of nearby replicated units.
  - A wide net that can cope with several characters at once even if they overlap.

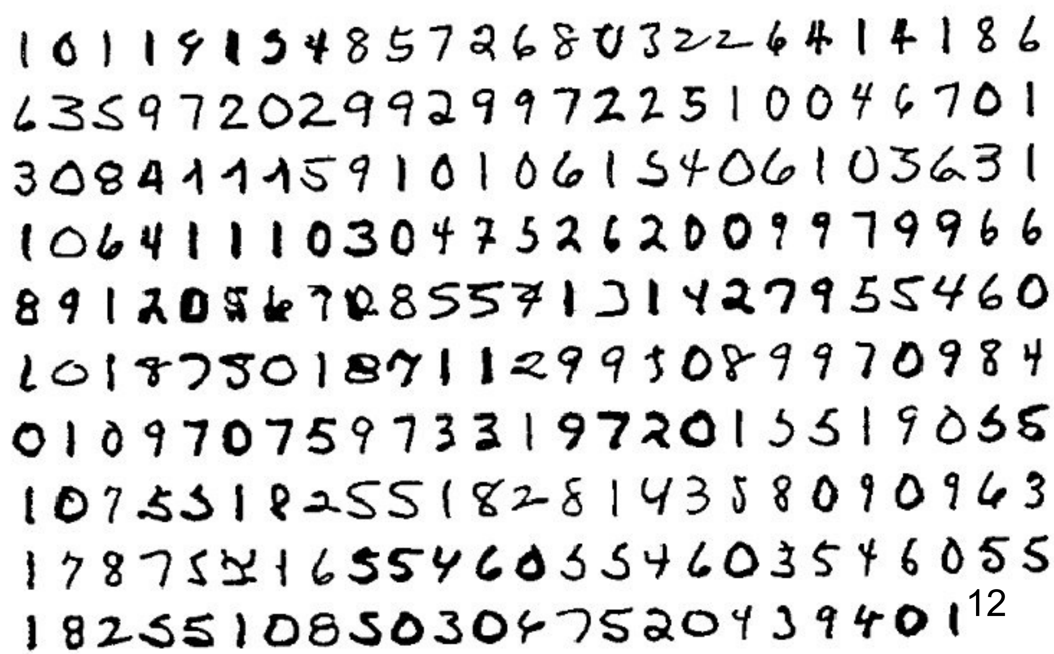
- Demo of LENET

Recognizing Digits

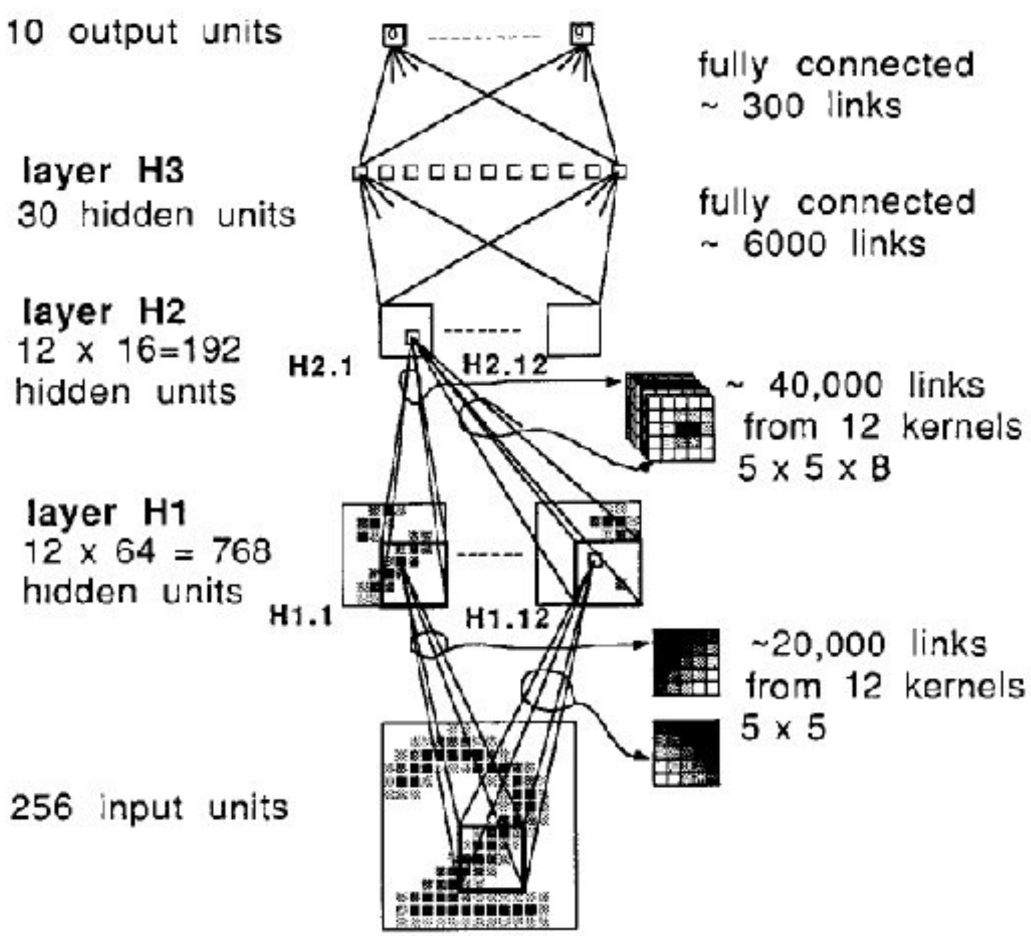
Hand-written digit recognition network

- 7291 training examples, 2007 test examples
- Both contain ambiguous and misclassified examples
- Input pre-processed (segmented, normalized)
  - 16x16 gray level [-1,1], 10 outputs





LeNet: Summary



Main ideas:

- Local → global processing
- Retain coarse posn info

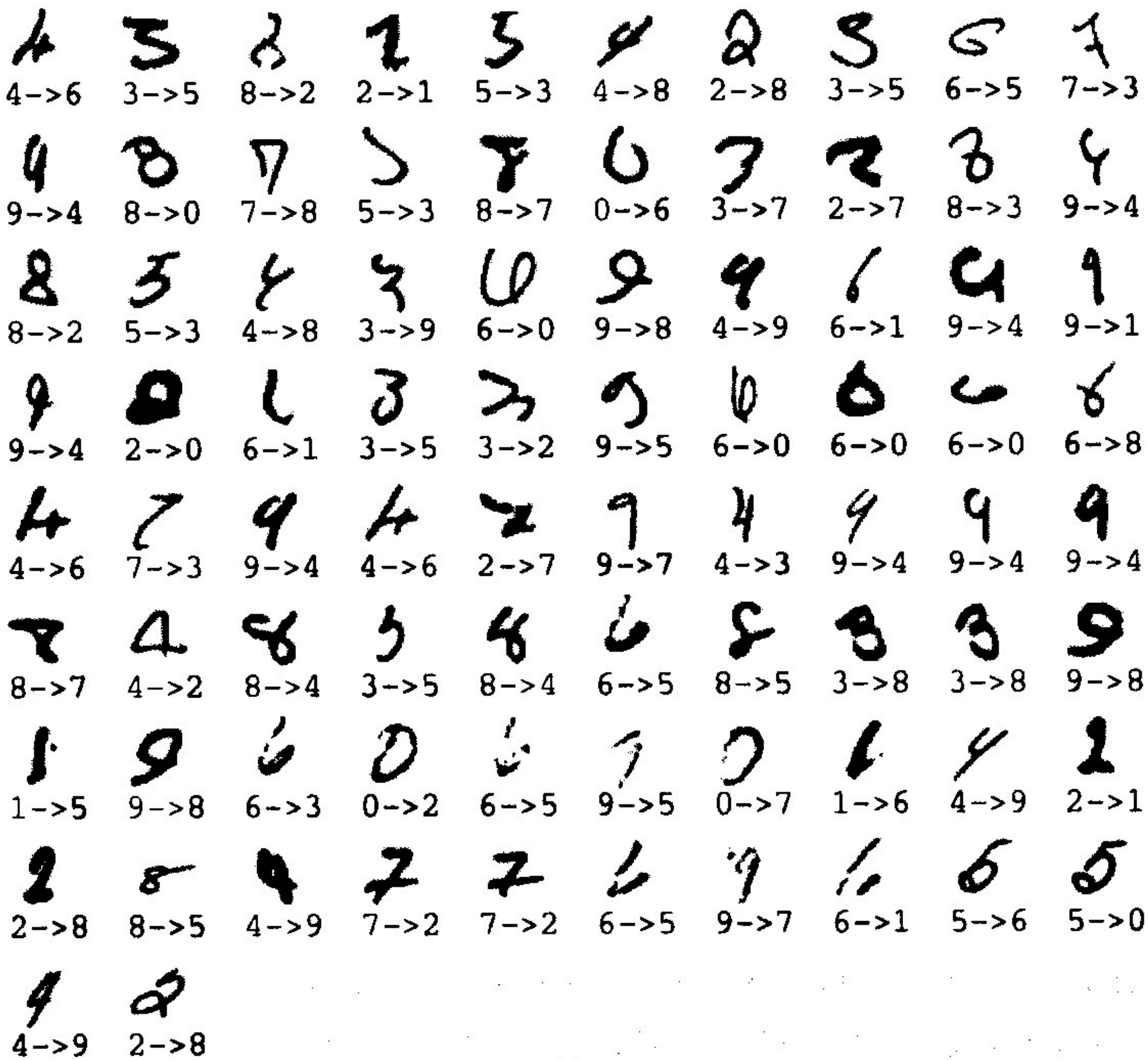
Main technique: weight sharing – units arranged in feature maps

Connections: 1256 units, 64,660 cxns, 9760 free parameters

Results: 0.14% (train), 5.0% (test)

vs. 3-layer net w/ 40 hidden units: 1.6% (train), 8.1% (test)

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

A brute force approach

- LeNet uses knowledge about the invariances to design:
  - the network architecture
  - or the weight constraints
  - or the types of feature

- But its much simpler to incorporate knowledge of invariances by just creating extra training data:
  - for each training image, produce new training data by applying all of the transformations we want to be insensitive to
  - Then train a large, dumb net on a fast computer.
  - This works surprisingly well

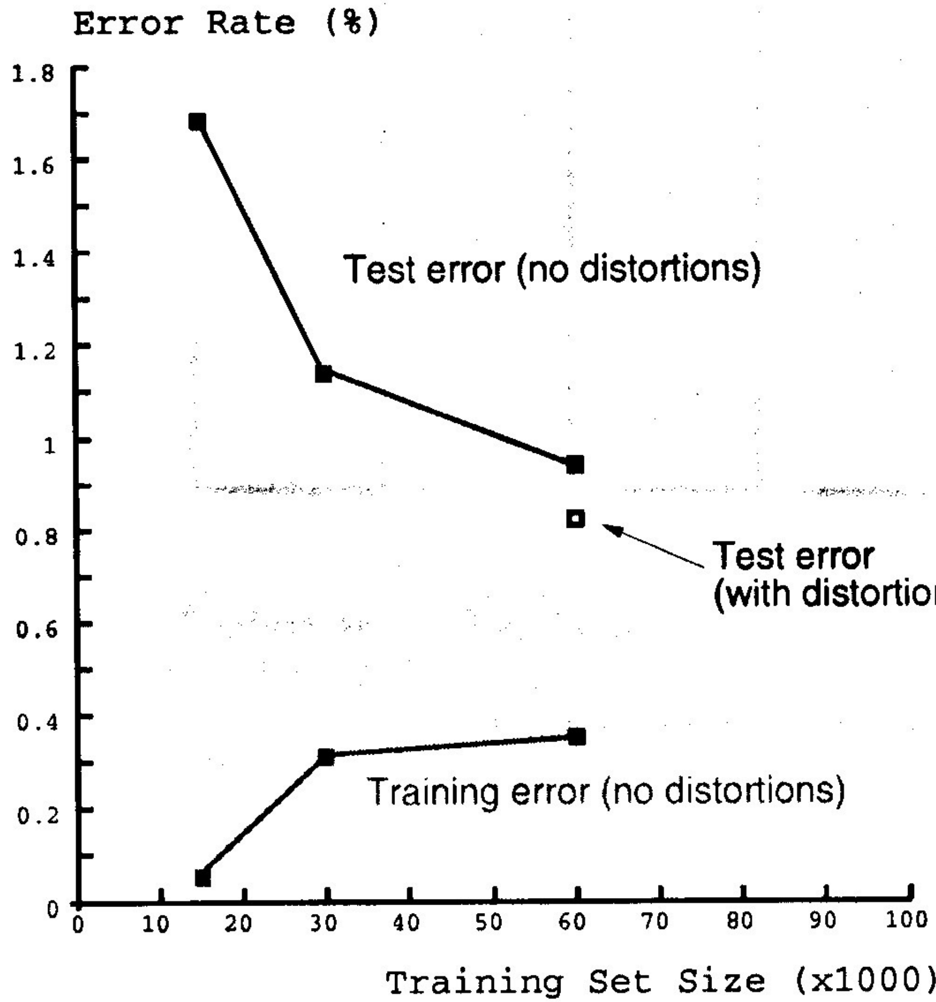


Fig. 6. Training and test errors of LeNet-5 achieved using training sets of various sizes. This graph suggests that a larger training set could improve the performance of LeNet-5. The hollow square show the test error when more training patterns are artificially generated using random distortions. The test patterns are not distorted.

Making backpropagation work for recognizing digits

- Using the standard viewing transformations, and local deformation fields to get lots of data.

- Use many, globally connected hidden layers and learn for a very long time
  - This requires a GPU board or a large cluster

- Use the appropriate error measure for multi-class categorization
  - Cross-entropy, with softmax activation

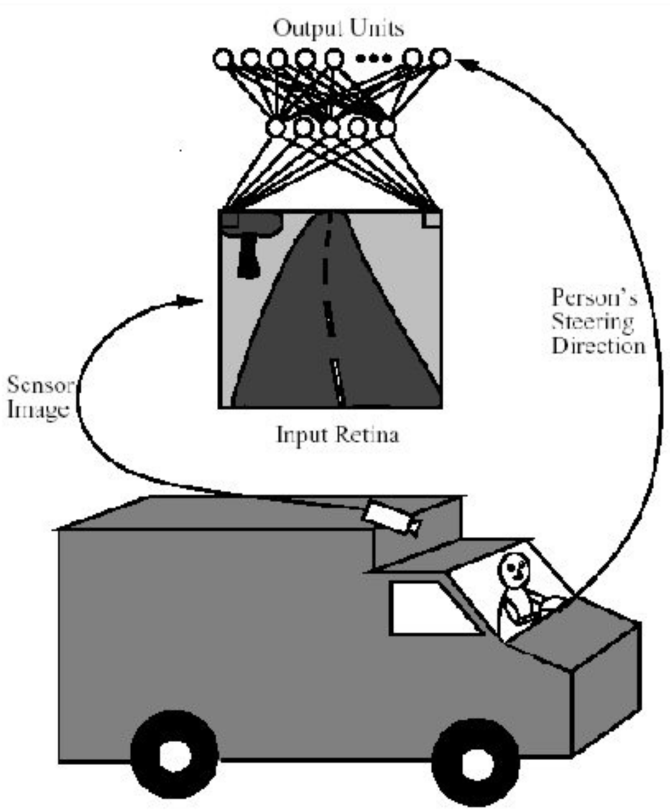
- This approach can get 35 errors on MNIST!

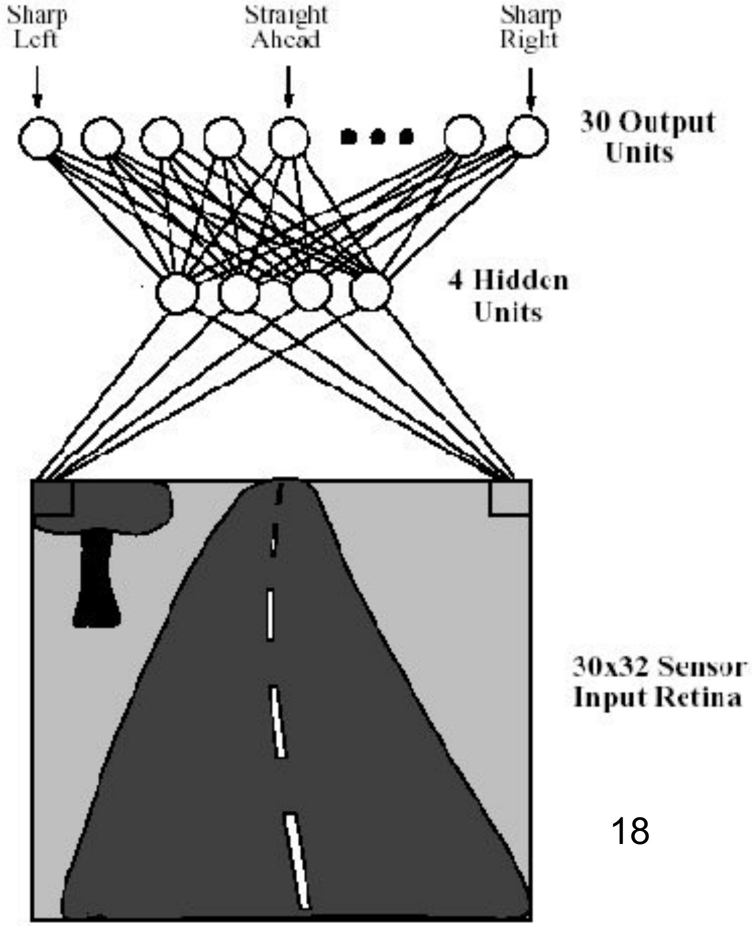
Fabricating training data

Good generalization requires lots of training data, including examples from all relevant input regions

Improve solution if good data can be constructed

Example: ALVINN



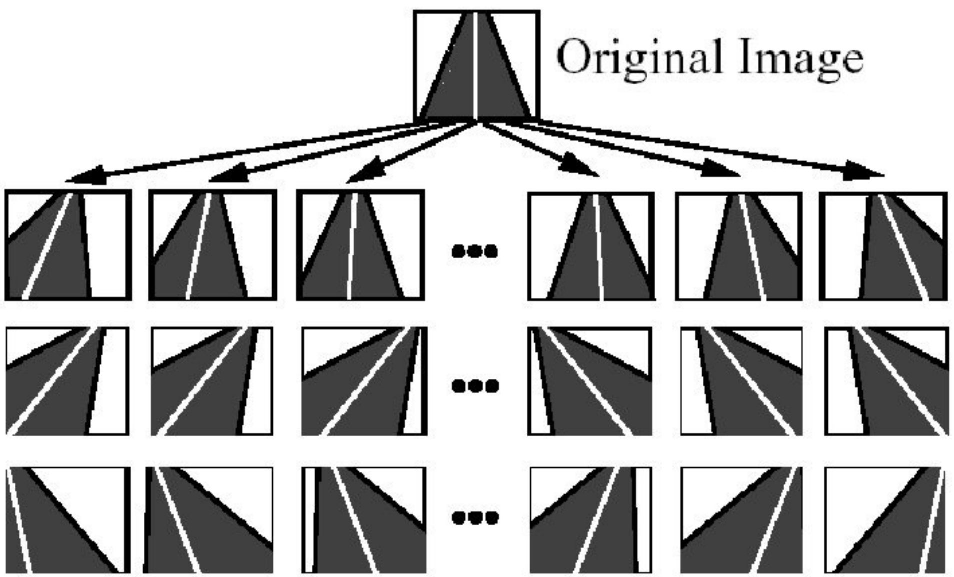


ALVINN: simulating training examples

On-the-fly training: current video camera image as input, current steering direction as target

But: over-train on same inputs; no experience going off-road

Method: generate new examples by shifting images



Shifted and Rotated Images

Replace 10 low-error & 5 random training examples with 15 new

Key: relation between input and output known!

Neural Net Demos

[Digit recognition]

[Scene recognition - Places MIT]

[Neural Nets Playground]

[Neural Style Transfer]