Deep Learning Examples

- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Digital assistants Google Now, Cortana
- Near-human-level autonomous driving

Neural Fuzzing

- Fuzzing is a popular technique for finding program vulnerabilities in complex software
  - Present the program with malicious input designed to crash it
  - Crafting malicious inputs is a complex problem
  - Input programs need to be mutated continuously
  - Potentially large number of mutations for a complex program
- Blackbox, Greybox and Whitebox Fuzzers
- Using Neural Network to predict “useful” mutations
- https://www.microsoft.com/en-us/security-risk-detection/

```
1  z = pow(3, a+b);
2  if(z < 1){
3    return 1;
4  }
5  else if(z < 2){
6    //vulnerability
7    return 2;
8  }
9  else if(z < 4){
10   return 4;
11 }
```

Reference:
Not all bytes are equal: Neural byte sieve for fuzzing
https://www.microsoft.com/en-us/research/wp-content/uploads/2017/11/neural-fuzzing-mcr.pdf

A recent quote from Andrew Ng

“ AI (Artificial Intelligence) technology is now poised to transform every industry, just as electricity did 100 years ago.

Between now and 2030, it will create an estimated $13 trillion of GDP growth.

While it has already created tremendous value in leading technology companies such as Google, Baidu, Microsoft and Facebook, much of the additional waves of value creation will go **beyond the software sector** “

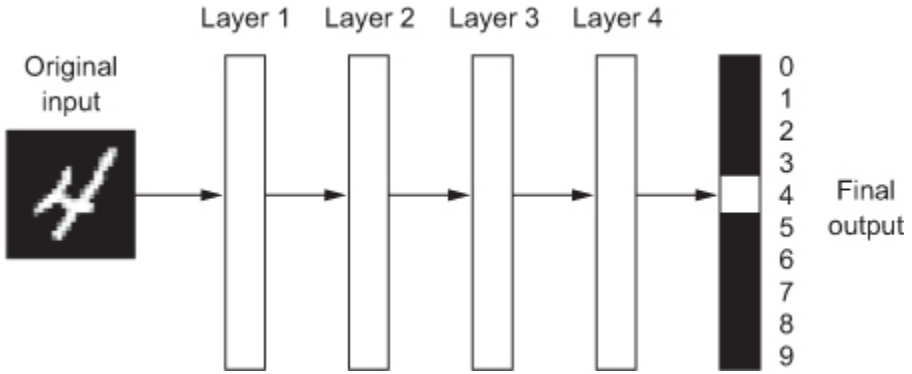
Why now? What about past AI winters?

- Hardware
- Datasets and benchmarks
- Algorithmic advances

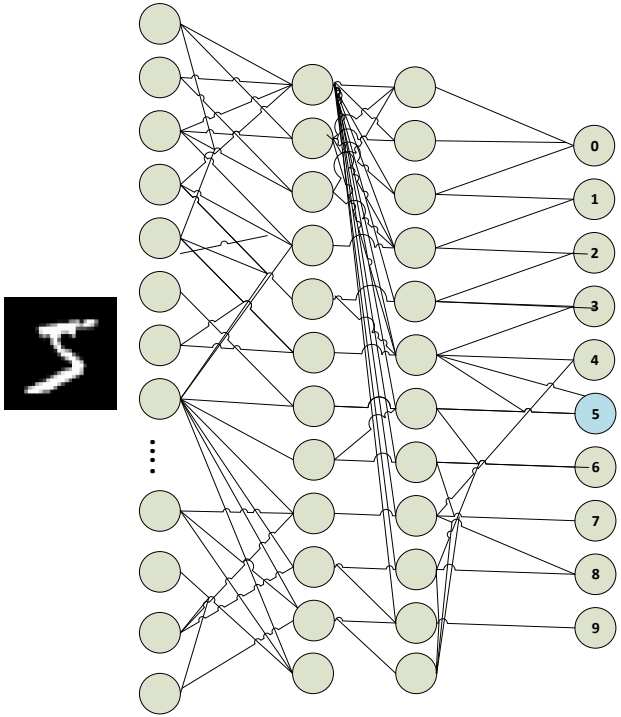
What about other ML techniques?

- *Probabilistic modeling (Naïve Bayes)*
- *Kernel Methods*
  - *Support Vector Machines*
  - *Decision Trees*
  - *Random Forests*
  - *Gradient Boosting Machines*

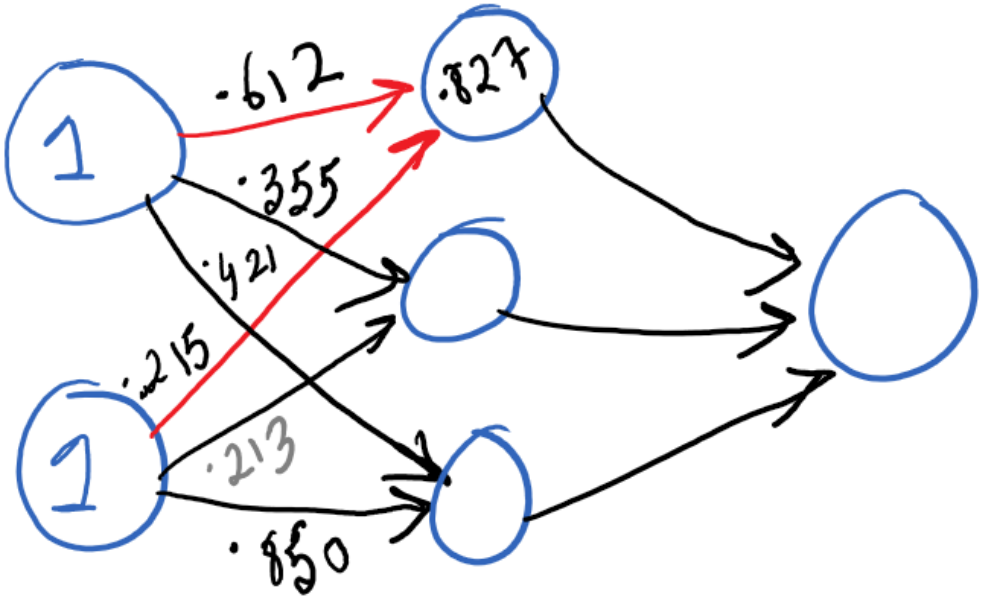
“Deep” in deep learning



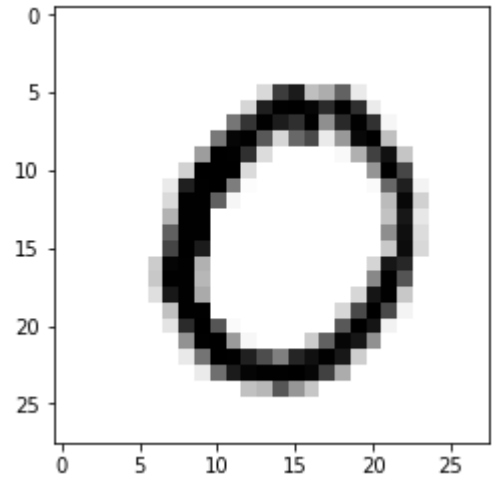
How can a network help us?

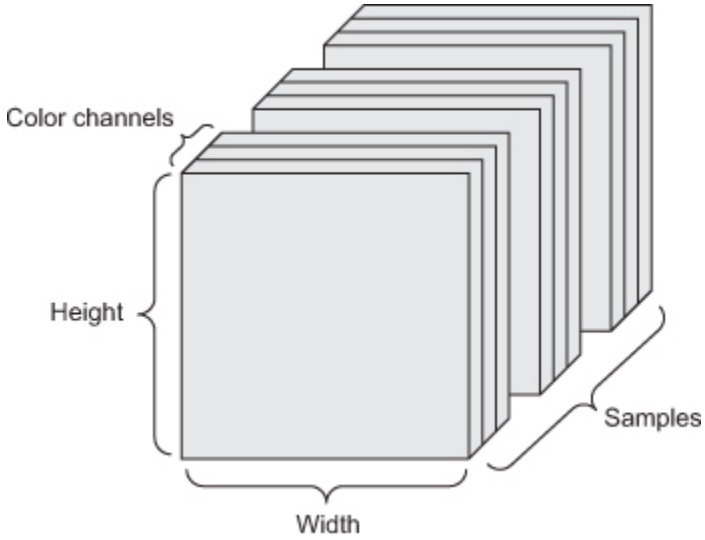


How does a network learn?



Tensor





shape (128, 256, 256, 1)

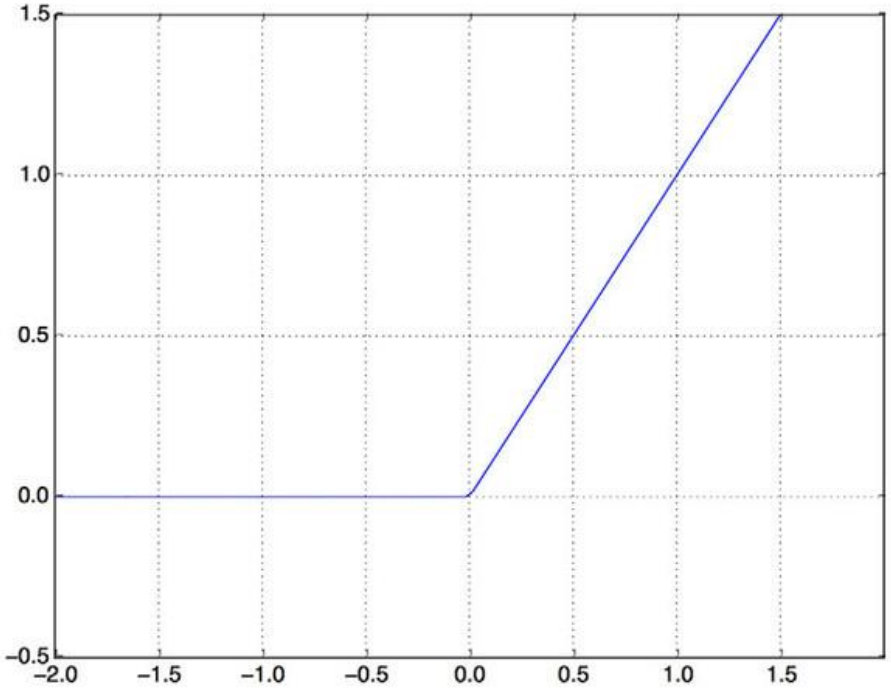
How layers transform data?

output = relu(dot(W, input) + b)

W- Weight

b - Bias

Rectified Linear Unit (RELU)

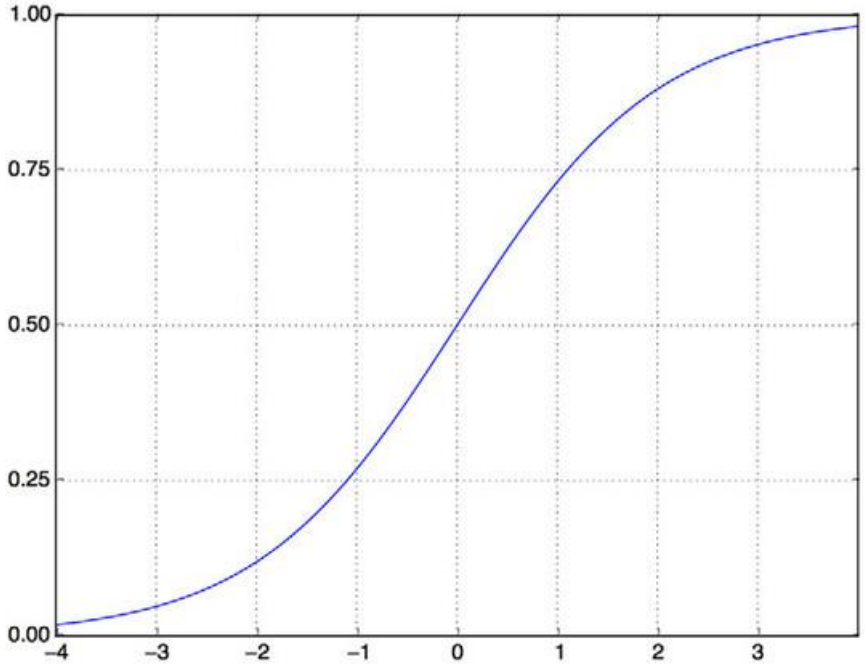


Power of parallelization

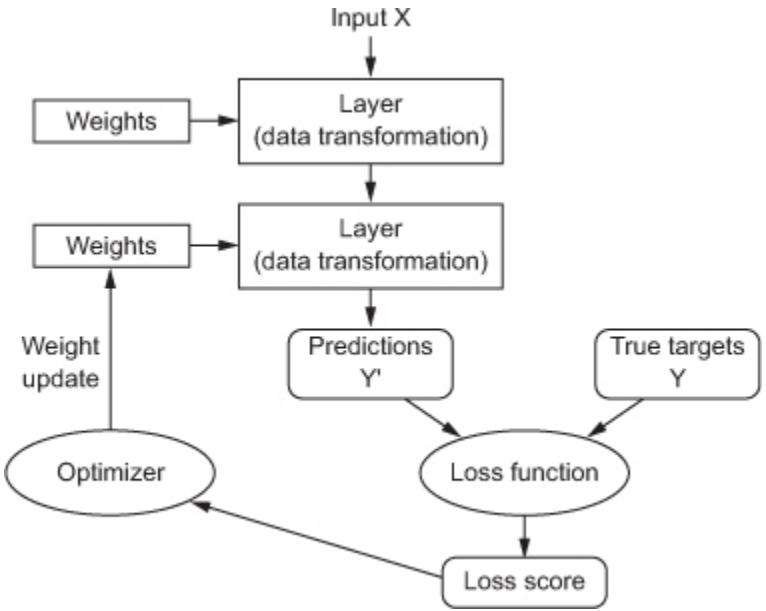
```
def naive_relu(x):
i in range(x.shape[0]): for j in
        range(x.shape[1]): x[i, j] = max(x[i, j], 0)
return x


z = np.maximum(z, 0.)
```

Sigmoid function



How network learns?

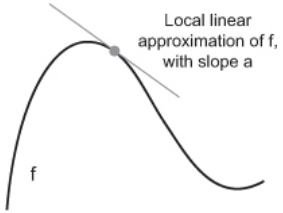


Training Loop

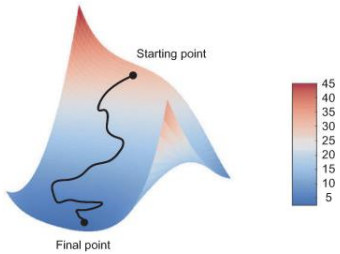
- Draw a batch of training samples x and corresponding targets y.
- Run the network on x
- Compute the loss of the network
- Update all weights of the network in a way that slightly reduces the loss on this batch.

Key Concepts

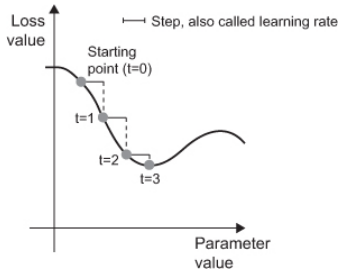
- Derivative



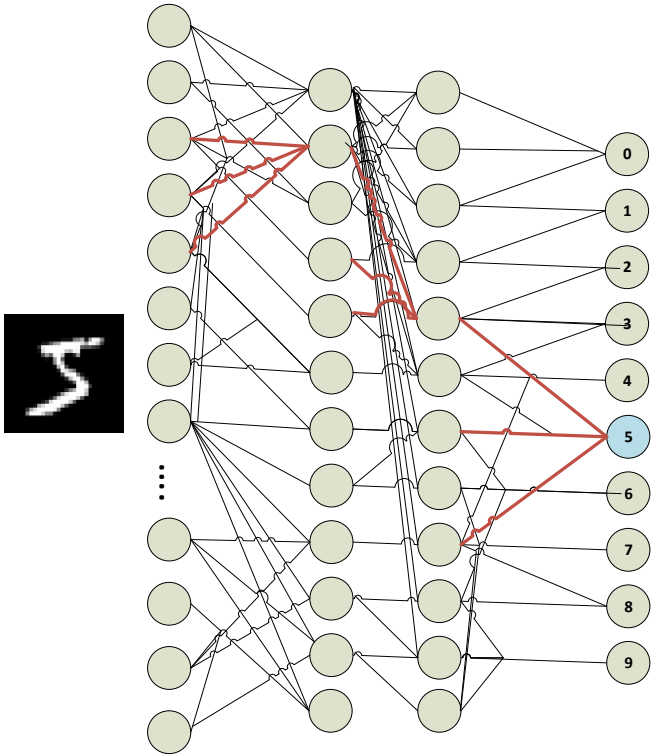
- Gradient



- Stochastic Gradient



Back propagation

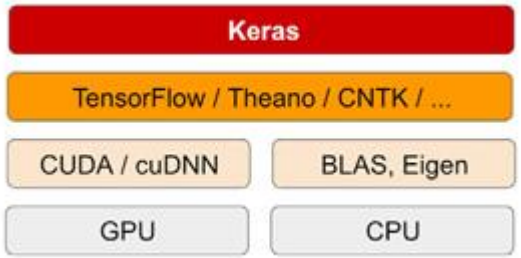


Network of layers

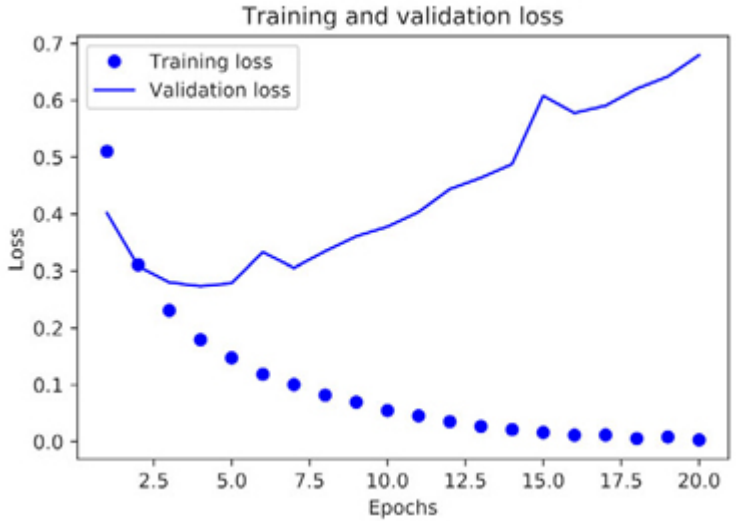
- Layers are like LEGO bricks of deep learning
- A data-processing module that takes as input one or more tensors and that outputs one or more tensors

Demo

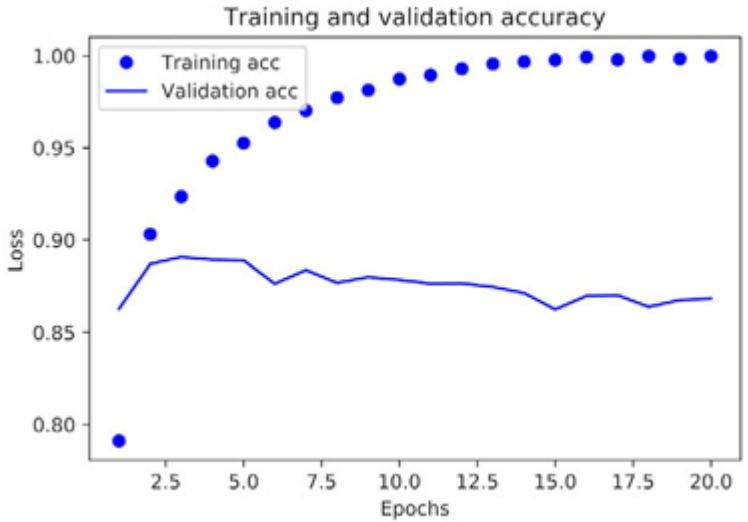
- Digit Classification
- Keras Library



Training and validation loss



Overfitting



Weight Regularization

- ***L1 regularization—*** The cost added is proportional to the *absolute value of the weight coefficients*
- ***L2 regularization—*** The cost added is proportional to the *square of the value of the weight coefficients*

Dropout

- Randomly *dropping out* (setting to zero) a number of output features of the layer during training.

Feature Engineering

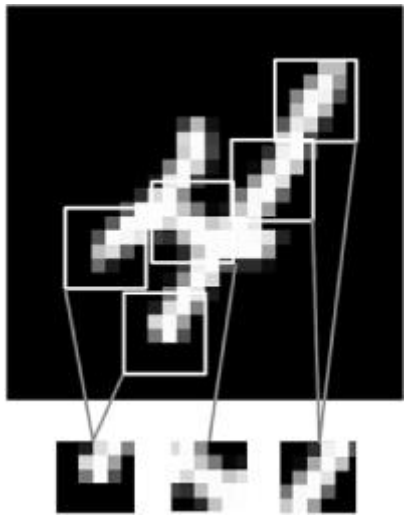
- process of using your own knowledge to make the algorithm work better by applying hardcoded (nonlearned) transformations to the data before it goes into the model

Feature engineering

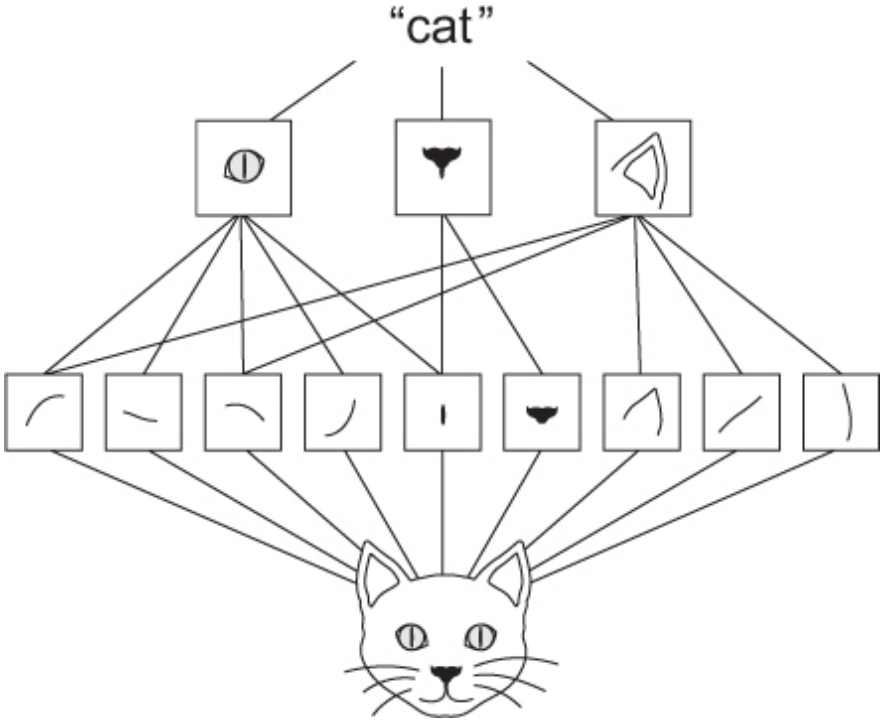
Raw data: pixel grid		
Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: -0.38, 2: 0.32}
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

ENTER CNN OR (CONVOLUTION NEURAL NETWORKS)

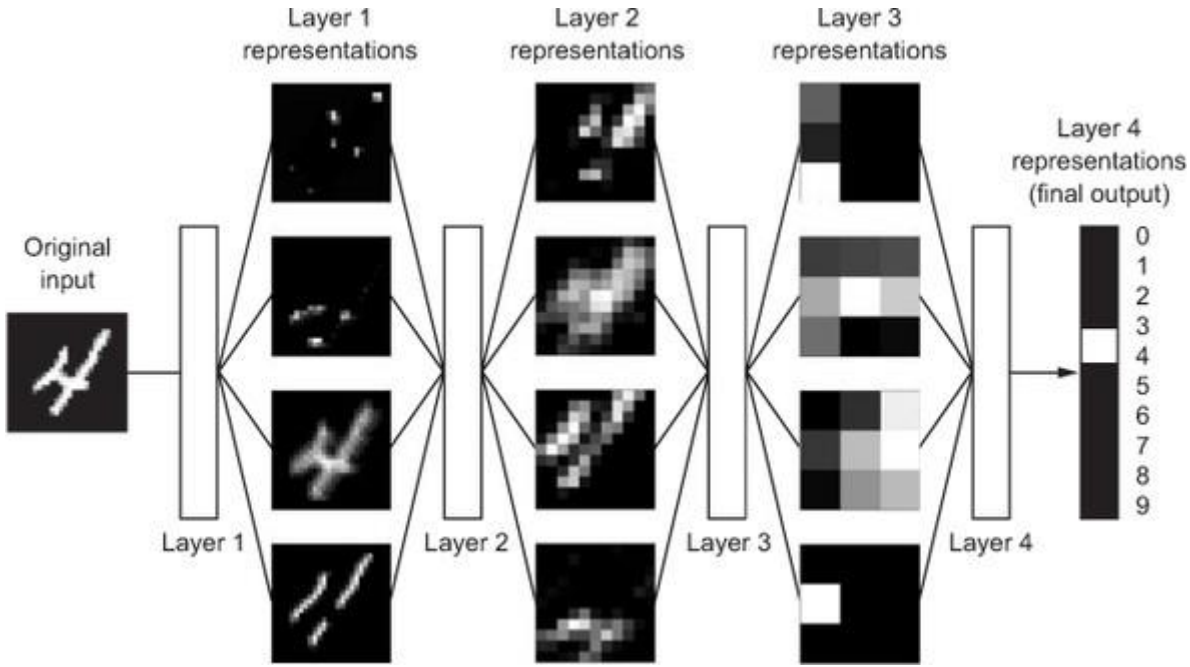
Convolution Operation



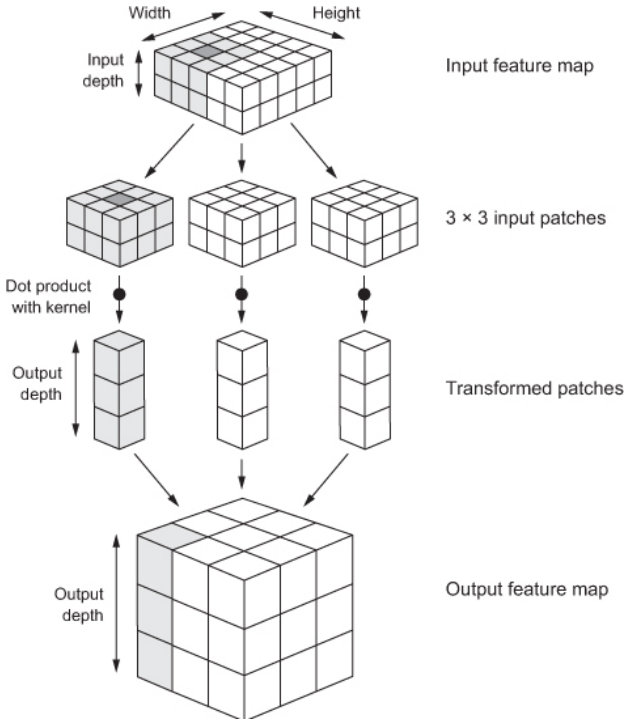
Spatial Hierarchy



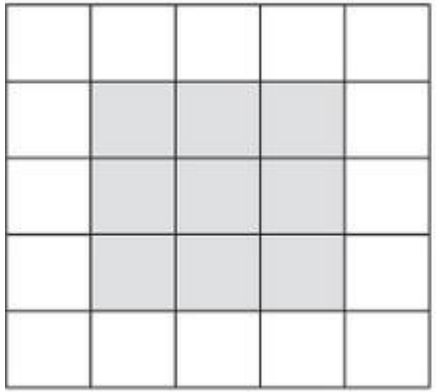
Digit Classification

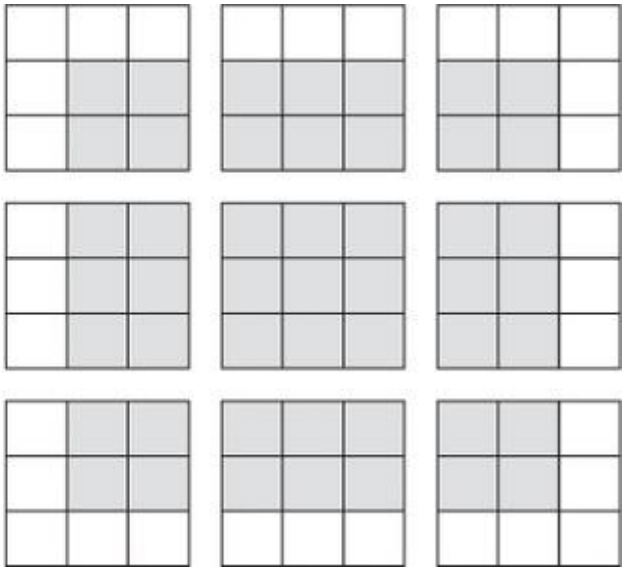


How Convolution works?

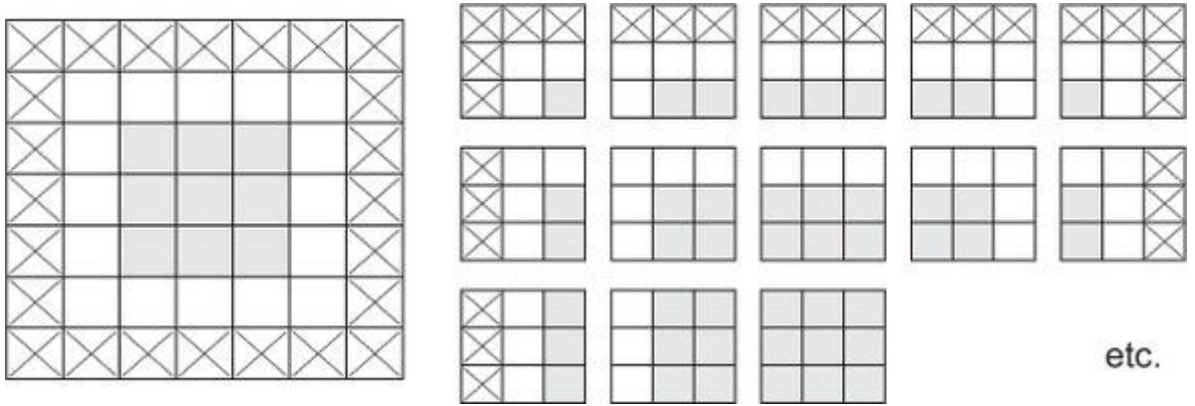


Sliding

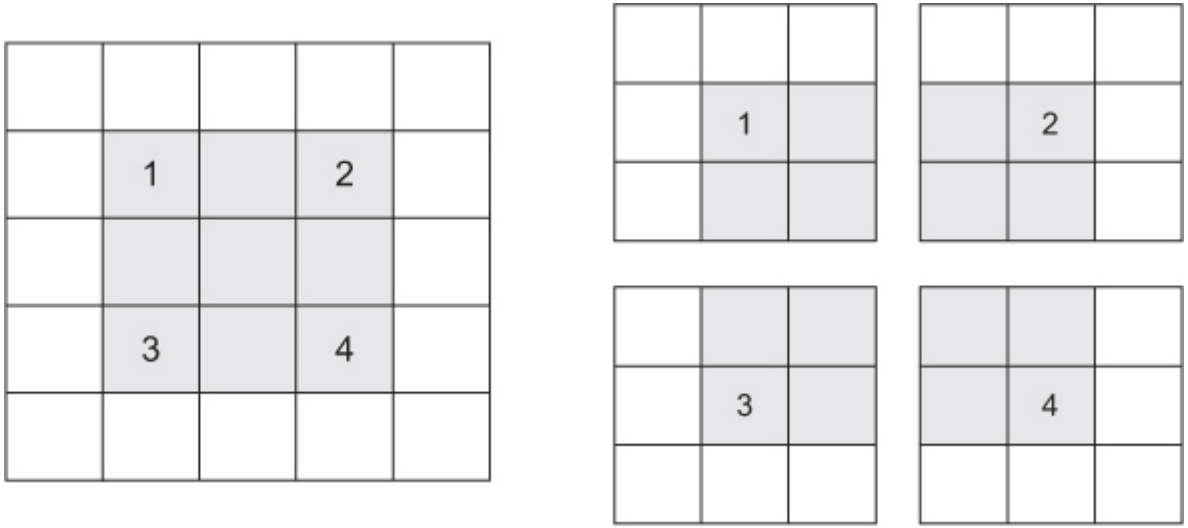




Padding

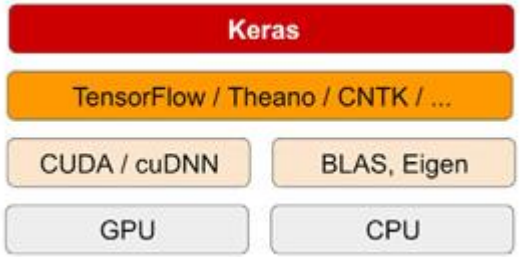


Stride



Demo

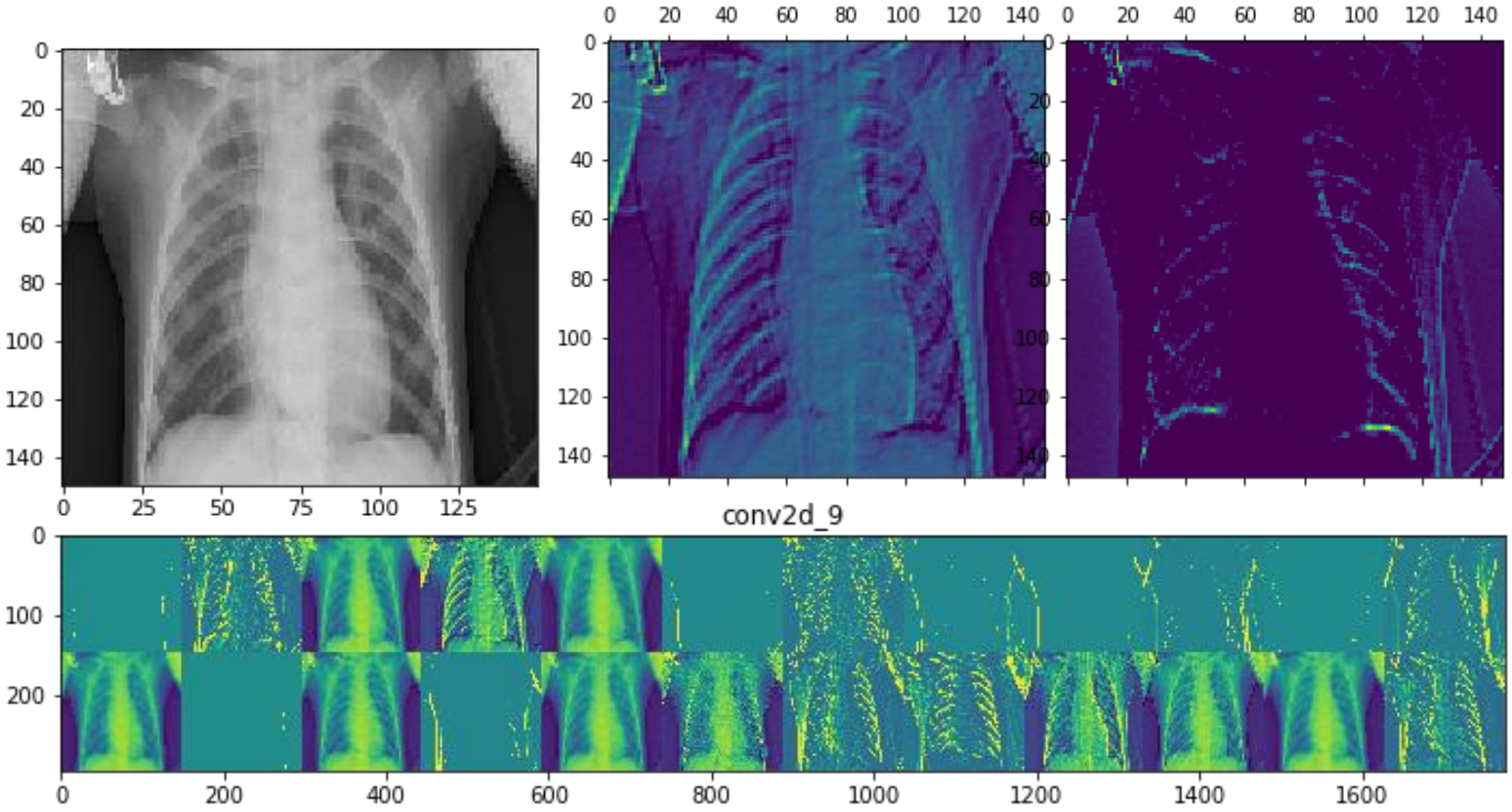
- Digit Classification (using CNN)
- Keras Library



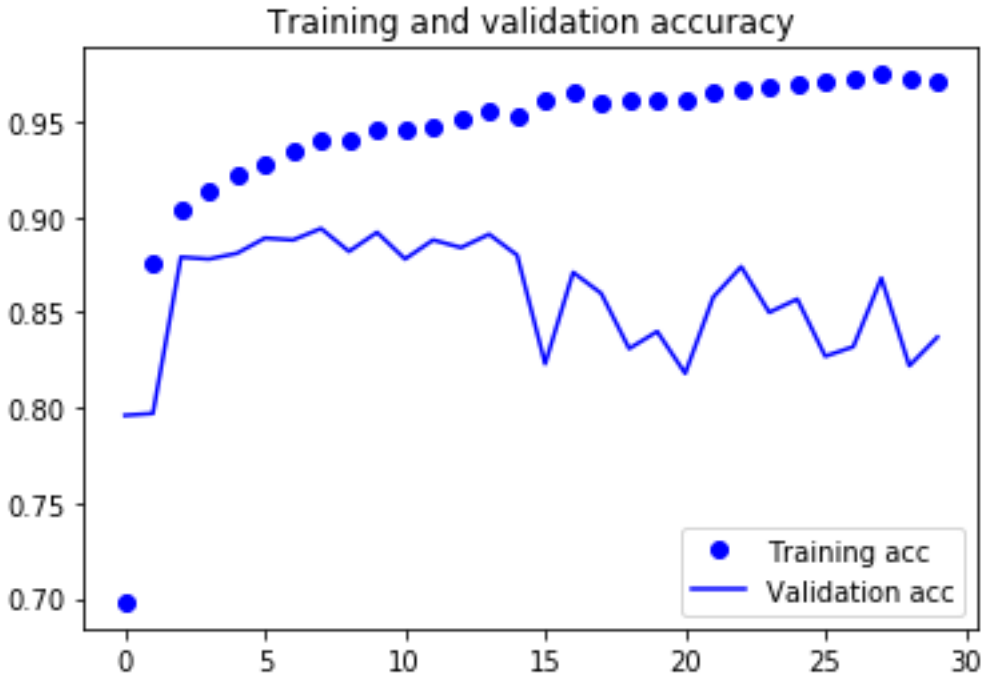
Demo

- X-ray Image Classification
- Dealing with overfitting
- Visualizing the learning

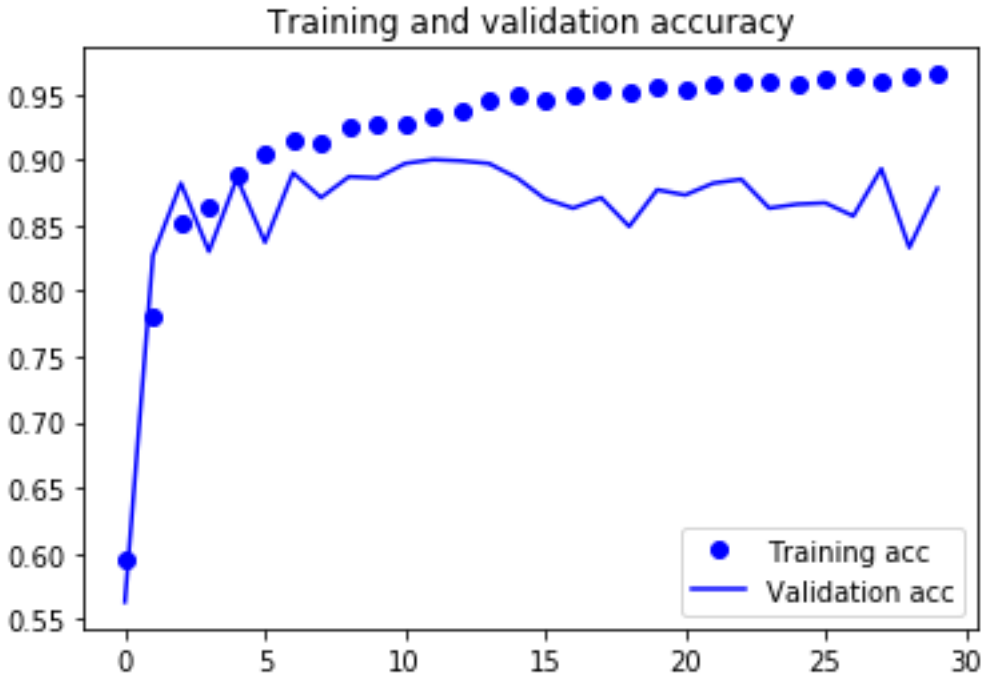
Visualizing the learning



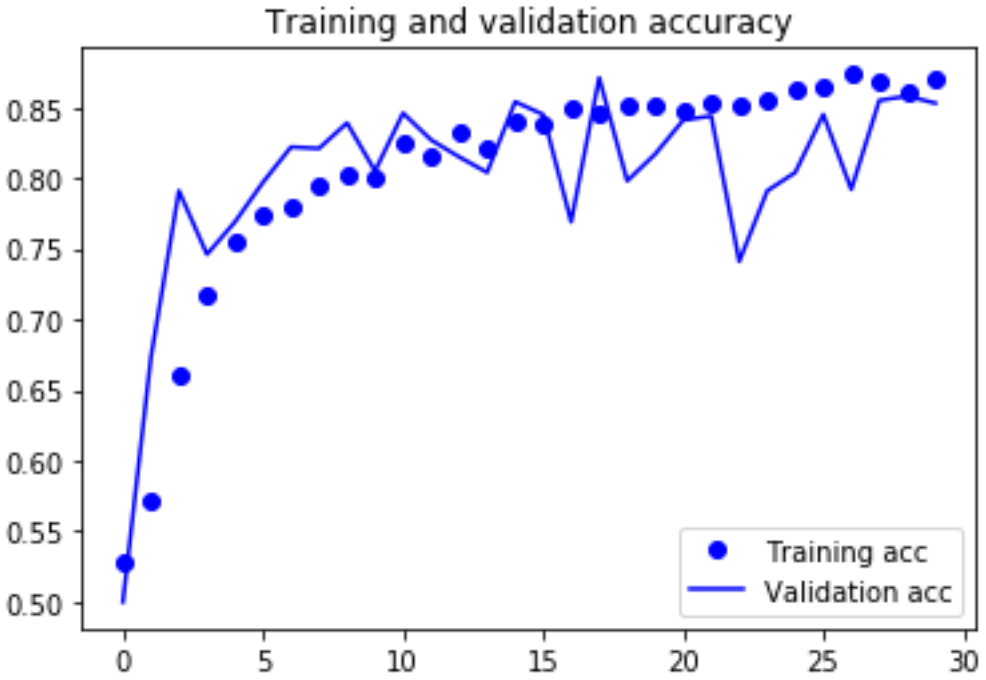
Overfitting



With dropout regularization



With data augmentation



Demo

- Activation heatmap

Stochastic gradient descent

- Draw a batch of training samples x and corresponding targets y.
- Run the network on x to obtain predictions y_pred.
- Compute the loss of the network on the batch, a measure of the mismatch between y_pred and y.

Stochastic gradient descent

- Compute the gradient of the loss with regard to the network's parameters (**a backward pass**).
- Move the parameters a little in the opposite direction from the gradient—for example W = step * gradient—thus reducing the loss on the batch a bit.