Original CNN Architecture

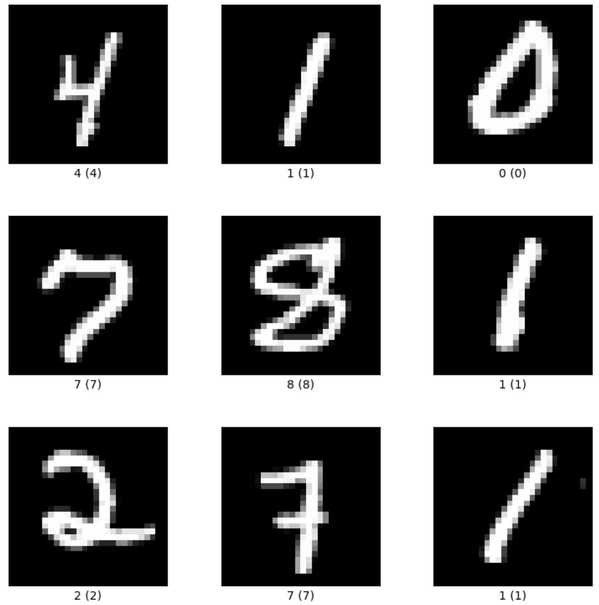
EE130: Distributed Machine Learning

Aryaman Pandya, 04/04/2022

Tufts UNIVERSITY | School of Engineering

Presentation Overview

- Introduction to the paper
  - Author spotlight
- Feature Engineering to Learning
- Convolutional Neural Networks
- Simulating LeNet-5
- SoA 23 years later
- References

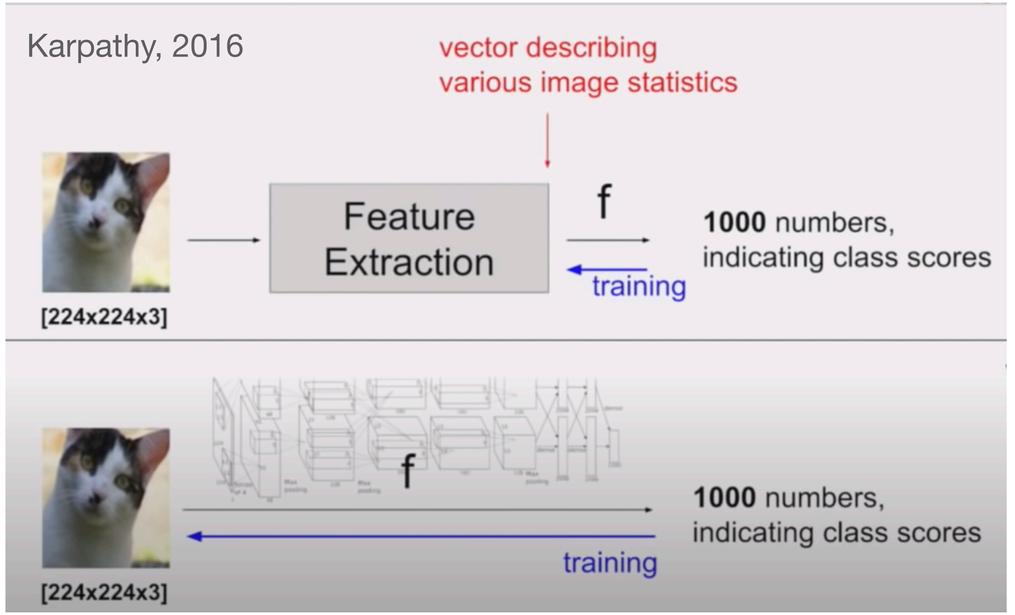


Tufts UNIVERSITY | School of Engineering

Object Recognition with Gradient Based Learning

Yann LeCun[1], Patrick Haffner, León Bottou, Yoshua Bengio[1]

- AT&T Shannon Labs, 1999
- Led the transition from feature engineering to learning
- Many believe this paper (and LeCun et. al.1989) to be the birth of deep learning
- Original network trained for 24+ hour
  - Computational inefficiencies led to stagnation of this innovation



[1]Turing Award winners

Tufts UNIVERSITY | School of Engineering

Author Spotlight

Yann LeCun

- Silver Professor of the Courant Institute of Mathematical Sciences, NYU
- Chief AI Scientist, Meta
- Turing Award winner, 2019
- Highly recommend: Lex Fridman podcast with Yann LeCun

Meta

NYU | COURANT

acm TURING AWARD

Tufts UNIVERSITY | School of Engineering

Gradient-Based Learning

Prior Architecture: Segmenter, Feature Extractor, Classifier

- We have an objective function, $Y^p = F(Z^p, W)$
  - Y represents the output for the p-th input
  - Z represents the p-th input
  - W is the set of weights assigned in the network
- We essentially want to reduce the loss function $E^p = \bar{D}(D^p, F(W, Z^p))$
  - D is the correct output for the p-th input
  - We want to reduce the average E over all training data, E_{train}
  - Originally used SGD

Tufts UNIVERSITY | School of Engineering

Gradient Based Learning

A little bit about the loss function

- What does our output look like?
  - Probability distribution: [0, 0.6, 0.2, 0.2]
  - Prediction values for different categories
- Categorical Cross-entropy

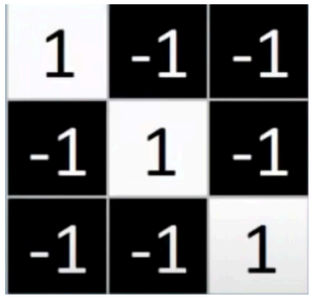
$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

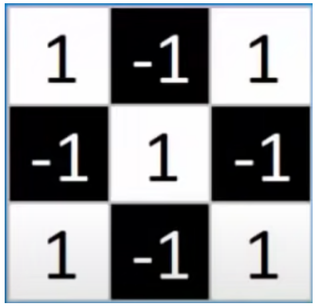
Tufts UNIVERSITY | School of Engineering

Convolutional Networks

Part 1: Convolutional layer

- Convolutional Filters of size a x b, where (a, b) < (n, m) if n, m are the input image dimensions
- Each convolutional layer is usually comprised of multiple filters testing for different features (edges, corners, lines etc.)
- Each filter is assigned a **weight** - this is where the learning happens





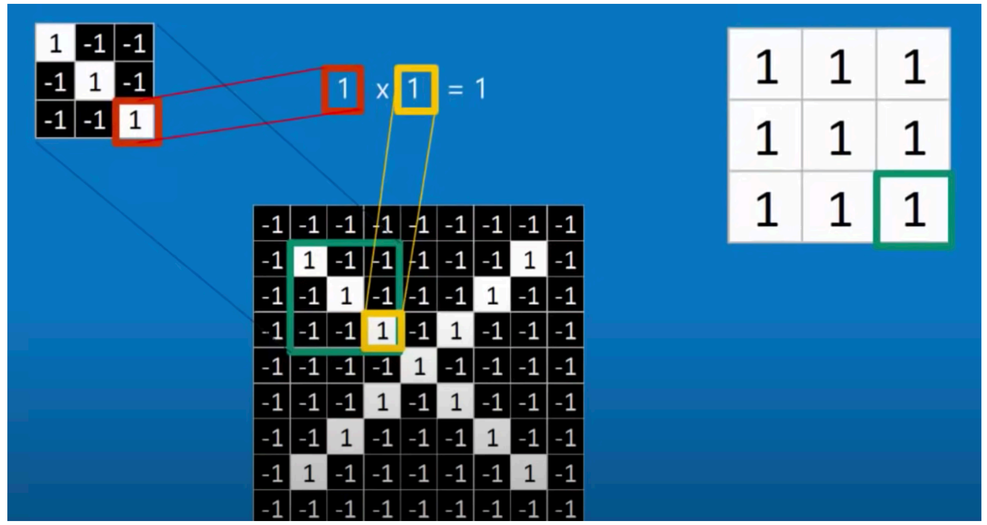
$$f(t) * g(t) := \underbrace{\int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau}_{(f*g)(t)},$$

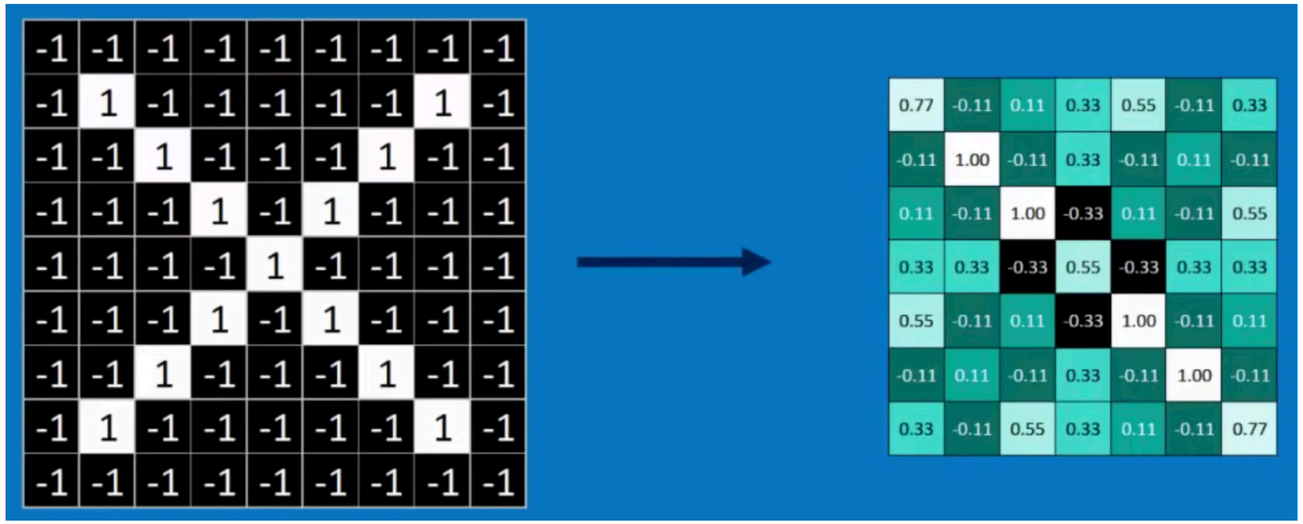
Tufts UNIVERSITY | School of Engineering

Convolutional Networks

Part 1: Convolutional layer

- We convolve the filters over pixels of the input, with some step size
- We do this over our entire input, giving us a “feature map”





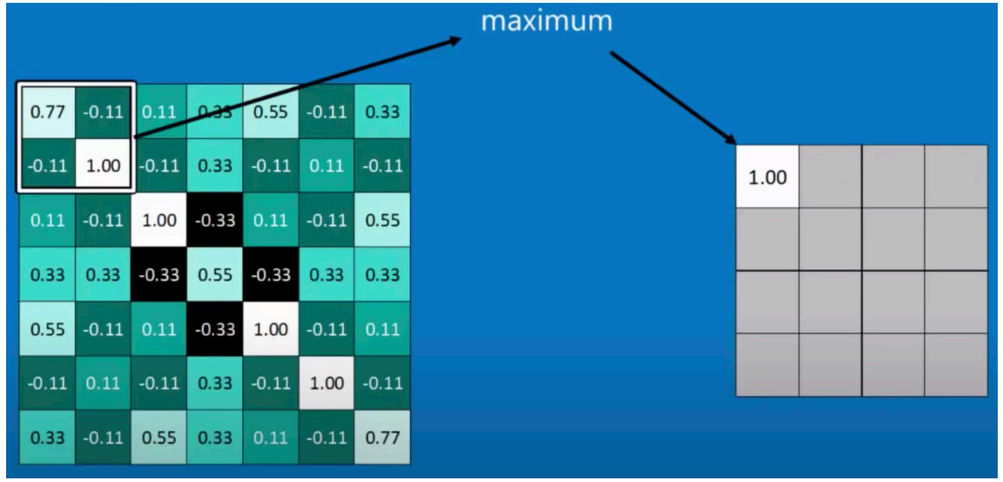
- Each convolutional layer is a stack of different filters, resulting in a stack of feature maps

Tufts UNIVERSITY | School of Engineering

Convolutional Networks

Part 2: Pooling Layer

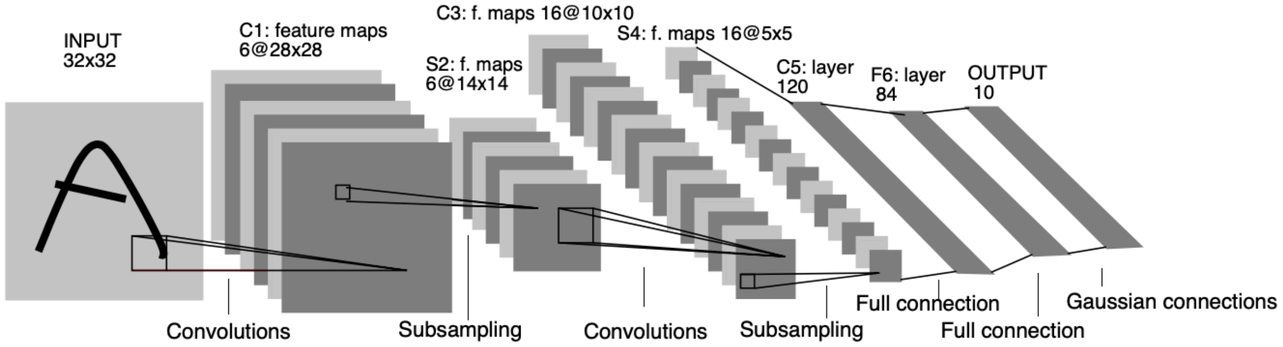
- Shrinking the image stack
- Often images are hundreds to thousands of pixels large
- Can get an accurate representation of what's in a collection of pixels by taking the maximum value pixel
- Increases time and computational efficiency
- **Improves accuracy with respect to transformations**



- Pooling is also done by taking the average of windows

Tufts UNIVERSITY | School of Engineering

LeNet-5 Architecture



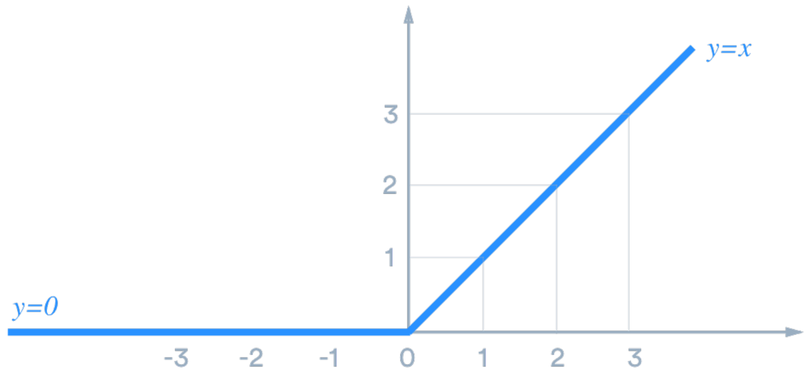
- The output from the final convolutional/pooling layer is flattened and fed to the fully connected layer
- After a few fully connected layers, we have a gaussian connection that allows us to convert to a probability distribution

Tufts UNIVERSITY | School of Engineering

Fully Connected Layers

- Neuron applies a linear transformation to an input vector
- Non-linear transformation function applied to the product
  - ReLU (Activation function)
- Gaussian Connection in order to convert to PDF
  - SoftMax function $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
    - (Activation function)

$$y_{jk}(x) = f\left(\sum_{i=1}^{n_H} w_{jk} x_i + w_{j0}\right)$$



Tufts UNIVERSITY | School of Engineering

LeNet-5 Simulation

- Using original dataset
- Model built using Keras, on top of tensorflow

Tufts UNIVERSITY | School of Engineering

23 Years Later

Improving the original model

- Can replace SGD with ADAM (better optimizers)
- Can handle larger datasets (ImageNet)
- We can increase the size of the model since we have the computational capability to do so
- Fundamentals remain the same
  - **Convolve, pool, normalize, classify**

Tufts UNIVERSITY | School of Engineering

References

- http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf
- https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy
- http://karpathy.github.io/2022/03/14/lecun1989/