Deep Learning I – Convolutional Neural Network

Dr Paul Yoo

Dept CSIS

28/11/19

4

Overview

We covered:

- Group Practical
- Application: IoT Intrusion Detection
- AWID (Kolias *et al* 2015)
- DEMISe (Parker *et al* 2019)
- Evaluation

We will cover:

- Convolutional Neural Network
  - Convolution in 2D and 3D
  - Non-linearity
  - Pooling
  - Classification
  - Other CNNs (Alexnet, VGG16)

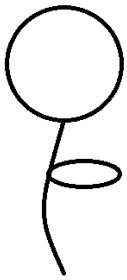
11

Consider a boy who's learning drawing and painting for the first time.

He has a dad who is not very good at art.

12

After a while, he draws this,



13

Let me show you a few more pictures

This time he draws something like this.

14

His dad repeats the process several times

Slowly and steadily he learns many things. From the basic skeleton to its fine features. Like:

- Shades and Shapes of the petals
- Edges of the leaves
- Features of the stem
- Etc

15

A CNN has two main components

Convolution filters + a full connected network

Convolutional Filters

- the boy learnt about the following
  - the round thing on the top or the stem and the leaf shape – very basic
  - a collection of similar looking units or petals – basic
  - the variation in shades – a bit more complex
- CFs capture low to high-level features of the image.

16

What are CNNs (a.k.a ConvNets)?

CNNs are a category of neural networks that have proven very effective in areas such as image recognition and classification.

ConvNets

- have been successful in identifying faces (e.g. Facebook), objects and traffic signs apart from powering vision in robots and self-driving cars.
- is able to recognise scenes and the system is able to suggest relevant captions ("a football player is kicking a football ball")
- is used for recognising everyday objects, humans and animals.
- have been effective in several NLP tasks (such as sentence classification) as well.

17

Source: https://arxiv.org/pdf/1506.01497v3.pdf

18

Gradient-Based Learning Applied to Document Recognition

// LeNet was one of the very first CNNs which helped propel the field of Deep Learning.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract—
Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.

Keywords— **Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.**

I. Introduction

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

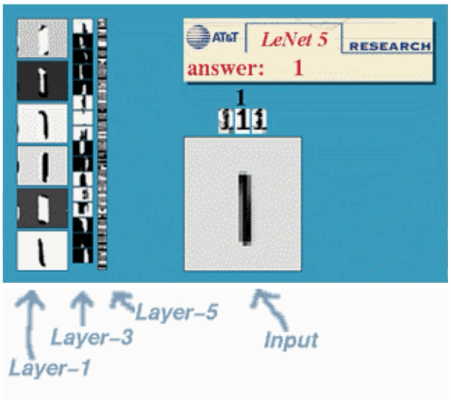
Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, November 1998

19

The LeNet Architecture (1990s)

This pioneering work by Yann LeCun was named LeNet-5 after many previous successful iterations since the year 1988.



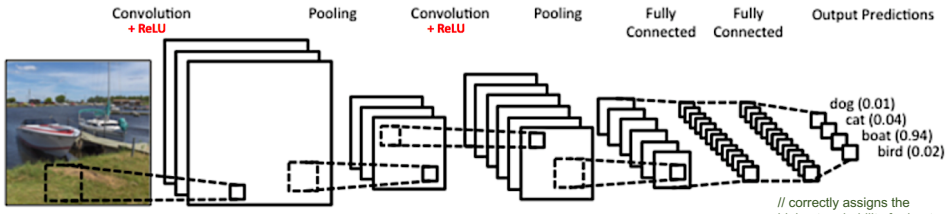
// the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc.

// trained on grey-scale images

20

CNN

There are four main operations (Conv., Non linearity, Pooling and Classification) – building blocks of CNN!



// this CNN architecture is similar in architecture to the original LeNet and classifies an input image into four categories: dog, cat, boat or bird

// correctly assigns the highest probability for boat (0.94) among all four categories.

21

Images

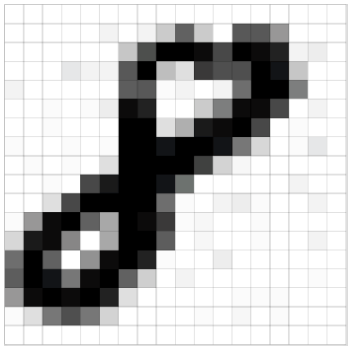
Every image can be represented as a matrix of pixel values.

Channel

- An image from a standard digital camera will have three channels – red, green and blue.
- Three 2d-matrices stacked over each other (one for each color), each having pixel values in the range 0 to 255.

A grayscale image has just one channel.

- A single 2d matrix representing an image.
- The value of each pixel in the matrix will range from 0 to 255 – zero indicating black and 255 indicating white.



Convolution

// Python: conv.forward TensorFlow: tf.nn.conv2d Keras: Conv2D

The purpose of Convolution is to extract features from the input image.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

// consider a 5 x 5 image whose pixel values are only 0 and 1

1	0	1
0	1	0
1	0	1

// consider another 3 x 3 matrix

// called '**filter**' or '**kernel**' or '**feature detector**'

// the convolution operation. the output matrix is called Convolved Feature or Feature Map.

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

// filter "sees" only a part of the input image in each stride.

4		

Convolved Feature

// called '**Activation Map**' or the '**Feature Map**'

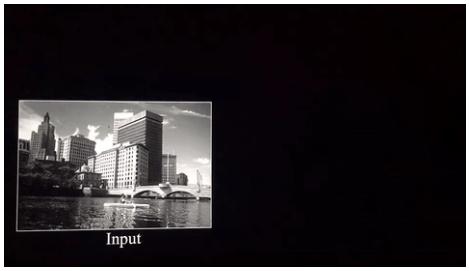
1. Slide the filter (orange matrix) over our original image (green) by 1 pixel (also called **'stride'**) and for every position
2. Compute element wise multiplication (between the two matrices)
3. Add the multiplication outputs to get the final integer which forms a single element of the output matrix (pink).

Filter

Different values of the filter matrix will produce different Feature Maps for the same input image.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

// different filters can detect different features from an image, for example edges, curves etc.



// two filters – one with red and green outlines.

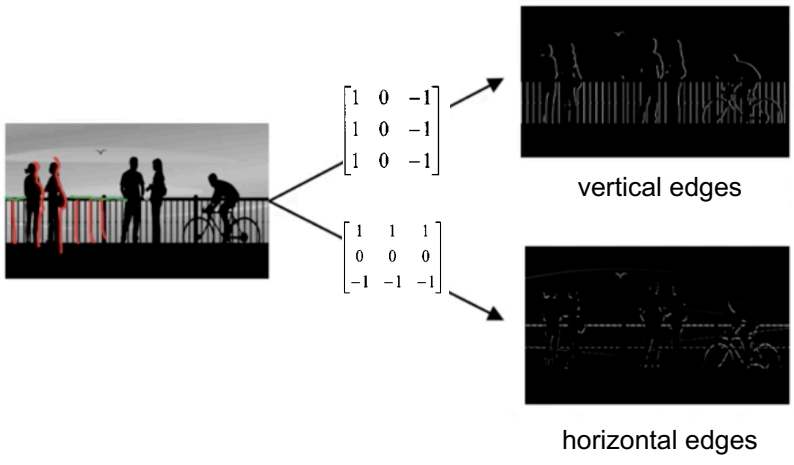
The Convolution Operation

// this captures the local dependencies in the original image.

Source: https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

24

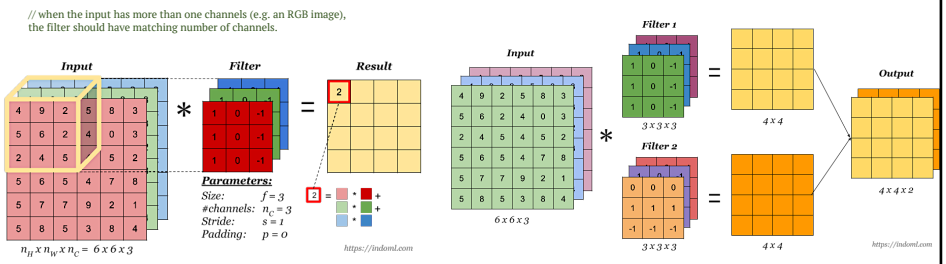
Vertical & horizontal edges



25

Convolution over volume

Multiple filters can be used to detect multiple features. # of output of the layer = # of filters in the layer.



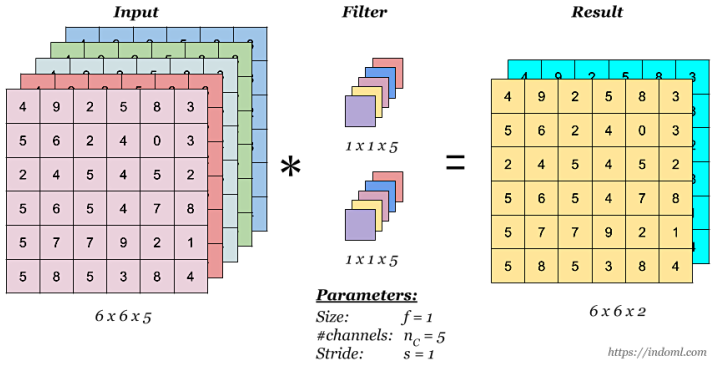
The total number of multiplications to calculate the result is (4 x 4) x (3 x 3 x 3) = 432

(4 x 4 x 2) x (3 x 3 x 3) = 864

26

1 x 1 Convolution

This is convolution with 1 x 1 filter. The effect is to flatten or "merge" channels together, which can save computations later in the network:



27

How to learn the values of filter?

- In practice, a CNN learns the values of these filters on its own during the training process
  - Specify parameters such as # of filters, filter size, architecture of the network etc. before the training process.
- The more # of filters we have, the more image features get extracted and the better our network becomes at recognising patterns in unseen images.

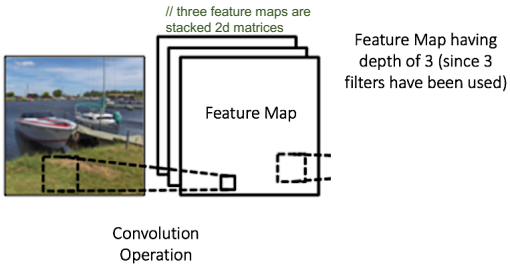
28

Feature Map

Decide three parameters the convolution step is performed (Depth, Stride and Padding)

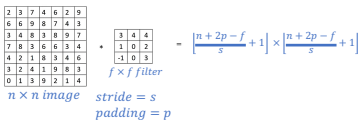
Depth corresponds to the # of filters we use for the convolution operation.

- Perform convolution of the original boat image using three distinct filters, thus producing three different feature maps as shown.



29

Feature Map cont.



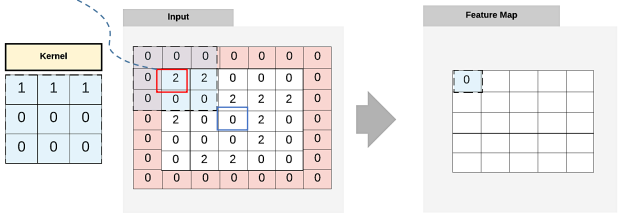
Stride is the number of pixels by which we slide our filter matrix over the input matrix.

- When the stride is 1 then we move the filters one pixel at a time.
- When the stride is 2, then the filters jump 2 pixels at a time as we slide them around.
- Having a larger stride will produce smaller feature maps.

Zero-padding.

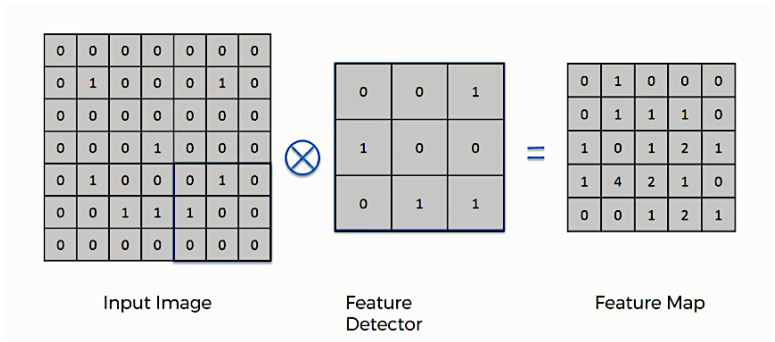
- Problems with **"valid" convolution** (no padding) // n*n x f*f → n− f+1 x n-f+1
  - Shrinking output.
  - Throwing away info from edge – the corner pixels used only one of the outputs.
- **"Same" Convolution** : Pad so that output size is the same as input size.
  - Pad the input with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.
  - Control the size of the feature maps. // n +2p - f +1 = n → p = (f-1)/2

=

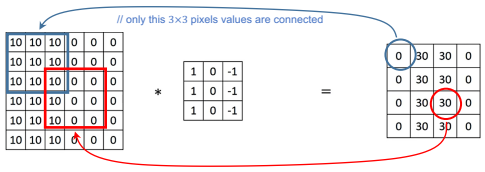


1. What is the value of stride? $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor = \left\lfloor \frac{7-3}{?} + 1 \right\rfloor = 5$
2. What are other names of feature detector?

Filter, kernel



Quiz



Why do we use the convolution operation?

- To extract features from the input image.
- Reduce # of parameters
  - Parameter sharing – feature detector that's useful in one part of the image is probably useful in another part of the image.
  - Sparsity of connections – in each layer, each output value depends only on a small number of inputs (less prone to overfitting)
- Reducing the size of the input image, the larger your strides, the smaller your feature map.
- Real images tend to be substantially larger and more complex than the 7 x 7 image – widen your strides.
- Easier to read.

Quiz

Do we lose information when using a feature detector?

- Yes – the feature map that we end up with has fewer cells and therefore less information than the original input image.
- However, the very purpose of the feature detector is to sift through the information in the input image and filter the parts that are integral to it and exclude the rest.
- It is meant to separate the wheat from the chaff.

Quiz

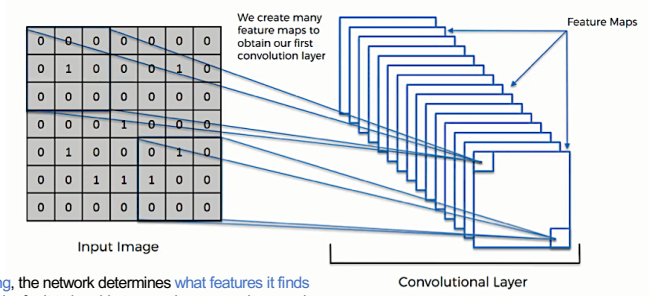
Why do we aim to reduce the input image to its essential features?

- What the convolution does is detect certain features
  - e.g. eyes and nose – you immediately know who you are looking at.
- These are all your brain needs to see in order to make its conclusion.
- CNNs operate in the similar way as your brain

34

Quiz

How to CNNs actually perform this operation?



// in reality, CNNs develop multiple feature detectors and use them to develop several feature maps which are referred to as convolutional layers.

- Through training, the network determines what features it finds important in order for it to be able to scan images and categorise them more accurately.
- Based on that, it develops its feature detectors.
- In many cases, the features considered by the network will be unnoticeable to the human eye, which is exactly why CNNs are so amazingly useful.

35

Quiz

What are other uses of convolution matrices?

- The word ”filter” is used in the same sense we use it when talking about Instagram filters.
- You can actually use a convolution matrix to adjust an image.

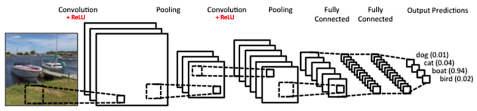
Sharpen:

Blur:

Edge Detect:

36

Non Linearity (ReLU)

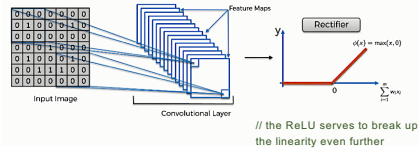


Break up the linearity in the image.

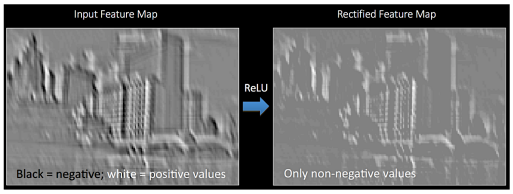
// convolution is a linear operatio – element wise matrix multiplication and addition

- e.g. the transition between pixels, the borders, the colours
- ReLU has been used after every cconvolution operation.
- ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

// **tanh** or **sigmoid** can also be used



// the ReLU serves to break up the linearity even further

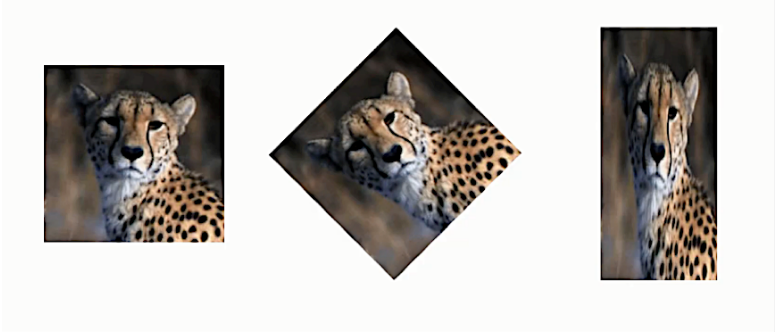


// remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).

37

Pooling

Enables the CNNs to detect the cheetah when presented with the image in any manner.



38

Pooling cont.

To teach your CNN to recognise that despite all of these differences, the network needs to acquire a property that is known as "spatial variance."

// the capability of detecting the object in the image without being confused by the differences in the image's textures, the distances from where they are shot or their angles.

// each posing differently in different settings and from different angles.

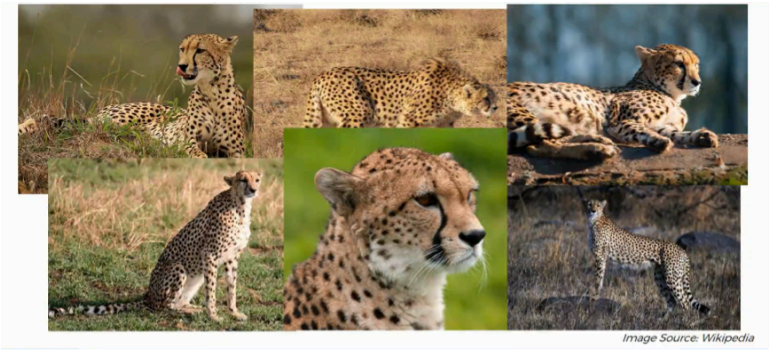


Image Source: Wikipedia

39

Pooling cont.

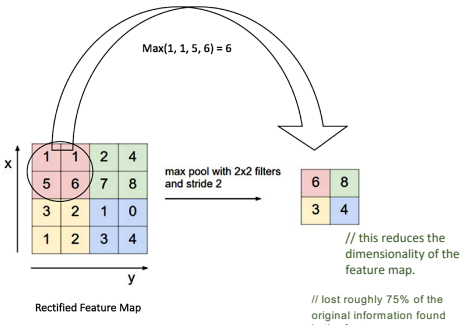
Spatial Pooling (a. k. a. subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information.

- Spatial Pooling can be of different types: Max, Average, Sum etc.
- In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window.
- In case of Average Pooling, instead of taking the largest element we could also take the average of all elements in that window.
- In practice, Max Pooling has been shown to work better.

Pooling cont.

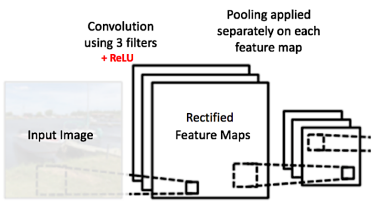
// generally, strides of two are most commonly used.

Slide 2×2 window by 2 cells (also called 'stride') and take the maximum value in each region. Two hyperparameters (f and s).



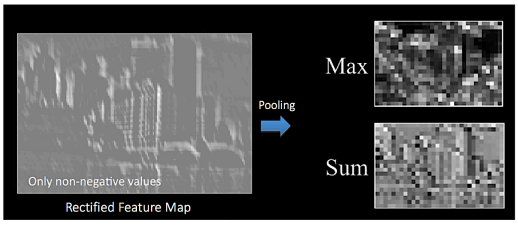
// this reduces the dimensionality of the feature map.

// lost roughly 75% of the original information found in the feature map



// pooling operation is applied separately to each feature map so we get three output maps.

Pooling cont.



- makes the input representations (feature dimension) smaller and more manageable.
- reduces the # of parameters and computations in the network, therefore, controlling overfitting.
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).
- helps arrive at an almost scale invariant representation of our image (the exact term is "equi-variant") – this is very powerful since we can detect objects in an image no matter where they are located.

42

Quiz

What does the pooling process provide CNN?

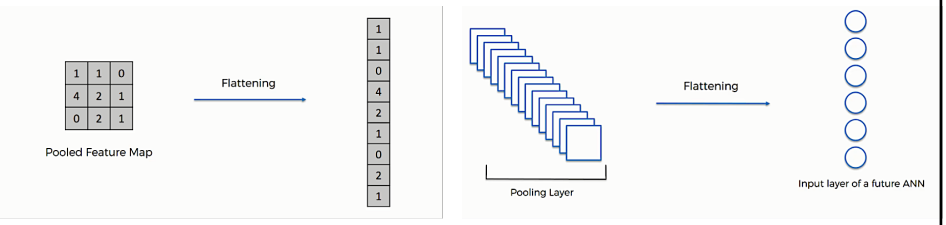
- Equivariant capability
- Speed up computation
- Reduce overfitting (abstract features could have less rules)

43

Flattening

We're going to need to insert this pooled feature map data into an ANN now.

- Flatten our pooled feature map into a column like in the image below.



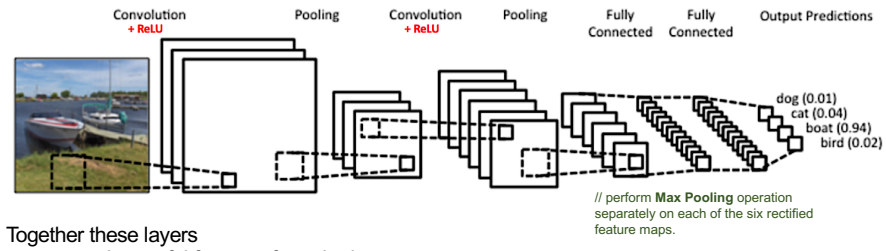
// after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

44

Three basic building blocks of CNN

// performs **Convolution** on the output of the first Pooling Layer using six filters to produce a total of six feature maps.

// **ReLU** is then applied individually on all of these six feature maps.



// perform **Max Pooling** operation separately on each of the six rectified feature maps.

Together these layers

- extract the useful features from the images,
- introduce non-linearity in the network and
- reduce feature dimension
- make the features somewhat equivariant to scale and translation

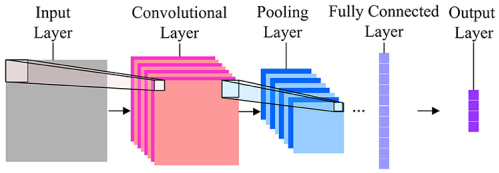
The output of the 2nd Pooling Layer acts as an input to the Fully Connected Layer.

45

Fully Connected Layer

The FCL is a traditional MLP that uses a softmax activation function in the output layer.

- The output from the convolutional and pooling layers represent high-level features of the input image.
- MLP uses these features for classifying the input image into various classes based on the training dataset.
- The sum of output probabilities from the Fully Connected Layer is 1.



46

Deep Learning using Linear Support Vector Machines

Yichuan Tang TANG@CS.TORONTO.EDU
Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

Abstract

Recently, fully-connected and convolutional neural networks have been trained to achieve state-of-the-art performance on a wide variety of tasks such as speech recognition, image classification, natural language processing, and bioinformatics. For classification tasks, most of these "deep learning" models employ the softmax activation function for prediction and minimize cross-entropy loss. In this paper, we demonstrate a small but consistent advantage of replacing the softmax layer with a linear support vector machine. Learning minimizes a margin-based loss instead of the cross-entropy loss. While there have been various combinations of neural nets and SVMs in prior art, our results using L2-SVMs show that by simply replacing softmax with linear SVMs gives significant gains on popular deep learning datasets MNIST, CIFAR-10, and the ICML 2013 Representation Learning Workshop's face expression recognition challenge.

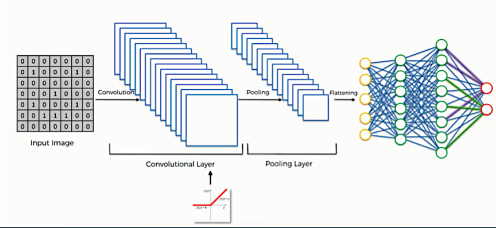
multistage process. In particular, a deep convolutional net is first trained using supervised/unsupervised objectives to learn good invariant hidden latent representations. The corresponding hidden variables of data samples are then treated as input and fed into linear (or kernel) SVMs (Huang & LeCun, 2006; Lee et al., 2009; Quoc et al., 2010; Coates et al., 2011). This technique usually improves performance but the drawback is that lower level features are not been fine-tuned w.r.t. the SVM's objective.

Other papers have also proposed similar models but with joint training of weights at lower layers using both standard neural nets as well as convolutional neural nets (Zhong & Ghosh, 2000; Collobert & Bengio, 2004; Nagi et al., 2012). In other related works, Weston et al. (2008) proposed a semi-supervised embedding algorithm for deep learning where the hinge loss is combined with the "contrastive loss" from siamese networks (Hadsell et al., 2006). Lower layer weights are learned using stochastic gradient descent. Vinyals et al. (2012) learns a recursive representation using linear SVMs at every layer, but without joint fine-tuning of the hidden representation.

47

A Quick Recap

- We start off with an input image.
- We apply filters (or feature maps) to the image, which gives us a convolutional layer.
- We then break up the linearity of that image using the rectifier function.
- The image becomes ready for the pooling step, the purpose of which is providing our CNN with the faculty of “spatial invariance”.
- After we're done with pooling, we end up with a pooled feature map.
- We then flatten our pooled feature map before inserting into an ANN.
- Throughout this entire process, the weights and the feature maps are trained and repeatedly altered in order for the network to reach the optimal performance.



48

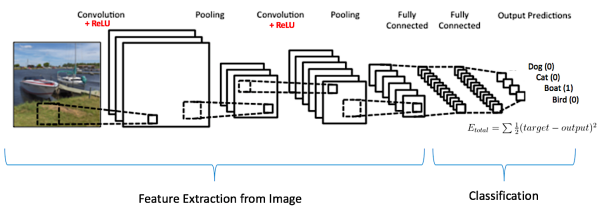
Putting it all together

Training using backpropagation

- Convolution + Pooling layers act as Feature Extractors from the input image while Fully Connected layer acts as a classifier.
- For example,

  Input Image = Boat

  Target Vector = [0, 0, 1, 0]



49

1. initialise all filters and parameters / weights with random values
2. the network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the FC layer) and finds the output probabilities for each class.
   a) lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
   b) since weights are randomly assigned for the first training example, output probabilities are also random.
3. calculate the total error at the output layer (summation over all 4 classes) – cross entropy or a mean squared error
   a) Total Error = $\sum$ ½ (target probability – output probability)²

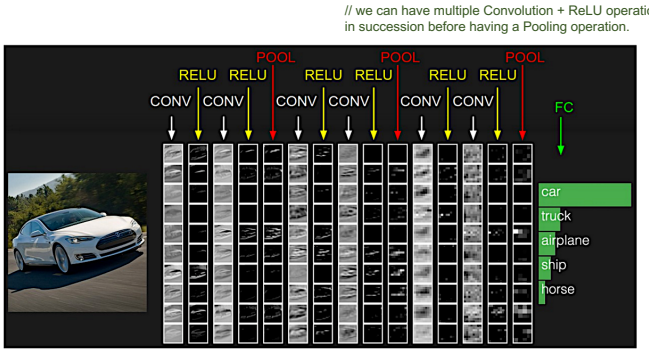
50

4. use backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use *gradient descent* to update all filter values / weights and parameter values to minimise the output error.
   a) the weights are adjusted in proportion to their contribution to the total error.
   b) when the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
   c) this means that the network has *learnt* to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
   d) parameters (# of filters, filter sizes, architecture of the network etc) have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.
5. repeat steps 2–4 with all images in the training set.

51

Convolution Design

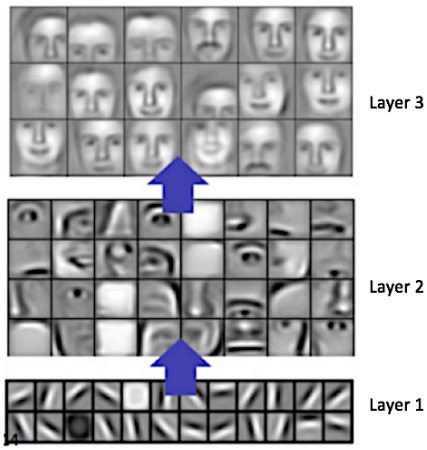
- In fact, some of the best performing CNNs today have tens of Convolution and Pooling layers!
- It is not necessary to have a Pooling layer after every Convolutional Layer.



VGG Net

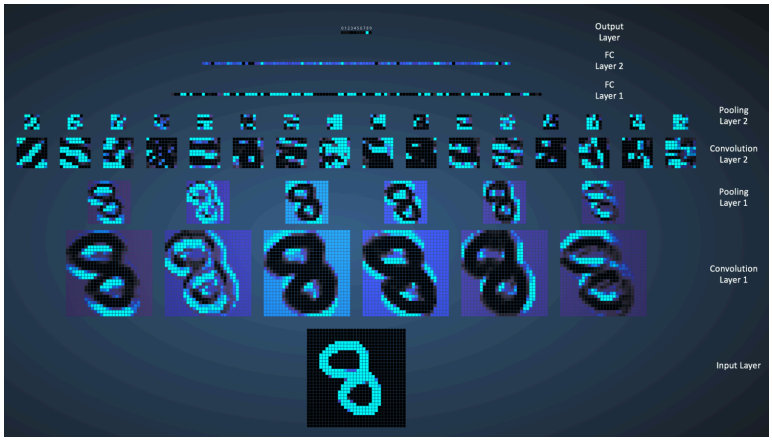
Convolution Steps

- In general, the more convolution steps we have, the more complicated features our network will be able to learn to recognize.
- For example,
  - 1st layer learns to detect edges from raw pixels,
  - 2nd layer uses the edges to detect simple shapes
  - 3rd layer uses these shapes to deter higher-level features such as facial shapes.



Visualisations of a CNN trained on the MNIST

Harley, A.W., 2015, December. An interactive node-link visualization of convolutional neural networks. In *International Symposium on Visual Computing* (pp. 867-877). Springer, Cham.



// 120 neurons in 1st layer, 100 neurons in 2nd layer, 10 neurons in 3rd layer corresponding to the 10 digits

// P2 does 2 × 2 max pooling (with stride 2).

// sixteen 5 × 5 (stride 1) convolutional filters
// pixel having the maximum value (the brightest one) in the 2 x 2 grid makes it to the Pooling layer.

// 2 × 2 max pooling (with stride 2)

// using six different filters produces a feature map of depth six.

// L1 is formed by convolution of six unique 5 × 5 (stride 1) filters

// input image contains 1024 pixels (32 x 32 image)

54

Other CNNs

CNNs have been around since early 1990s.

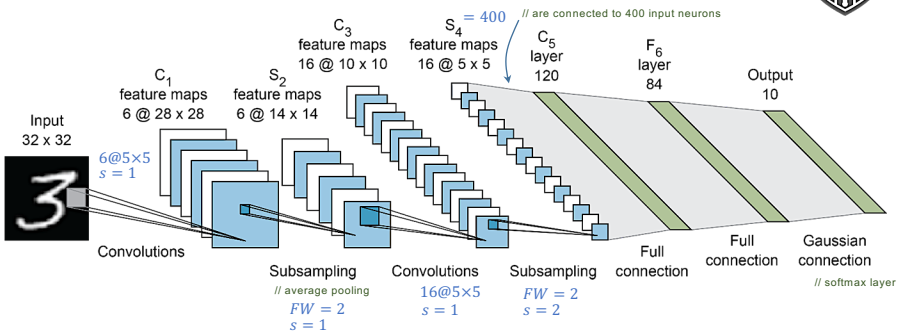
LeNet-5 (1990s)

- This pioneering work by Yann LeCun was named LeNet5 after many previous successful iterations since the year 1988!
  - There was no GPU to help training, and even CPUs were slow.
  - CNNs use sequence of 3 layers: convolution, pooling, non-linearity – this may be the key feature of Deep Learning for images since this paper!
  - use convolution to extract spatial features
  - subsample using spatial average of maps
  - non-linearity in the form of *tanh* or *sigmoids*
  - MLP as final classifier
  - sparse connection matrix between layers to avoid large computational cost

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

55

LeNet-5 Architecture



- 60K parameters Vs 10–100M parameters (today)
- As you go from left to right n_h and n_w decreases while n_c increases.
- Conv–Pool–Conv–Pool–fc–fc–output (typical architecture)
- Sigmoid/tanh (not ReLU)
- Read sections II and III

56

Gradient-Based Learning Applied to Document Recognition

// LeNet was one of the very first CNNs which helped propel the field of Deep Learning.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract—
Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.

Keywords— **Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.**

I. Introduction

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, November 1998

57

Other CNNs

1990s to 2012

// Google started in 1998
// Facebook started in 2004

- In the years from late 1990s to early 2010s CNNs were in incubation.
- As more and more data and computing power became available, tasks that CNNs could tackle became more and more interesting.

58

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

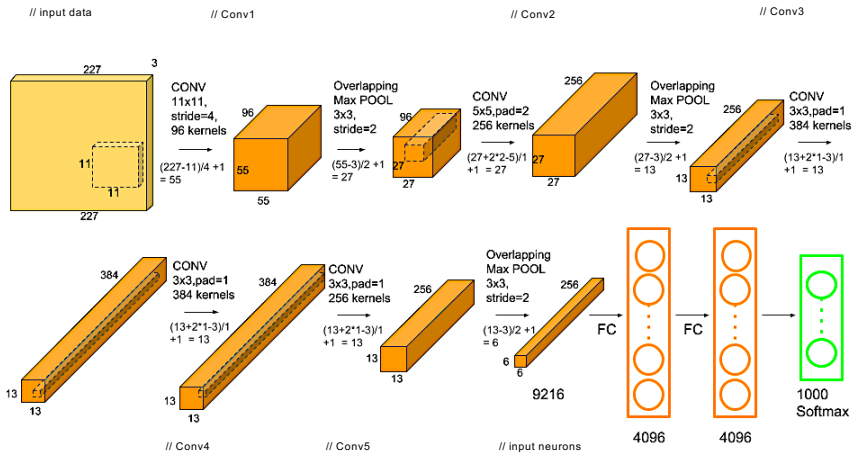
59

Other CNNs

AlexNet (2012)

- It was a significant breakthrough with respect to the previous approaches and the current widespread application of CNNs can be attributed to this work.
- use of ReLU as non-linearities
- use of dropout technique to selectively ignore single neurons during training, a way to avoid overfitting of the model
- max pooling, avoiding the averaging effects of average pooling
- use of multiple GPUs NVIDIA GTX 580 to reduce training time – $10x$ faster training time
- Local Response Normalisation – not very useful!
- Huge impact even beyond computer vision!
- Easier one to read.

AlexNet Architecture (5 Conv and 3 FC layers)



Lots of similarities but much bigger (60M parameters) !

Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

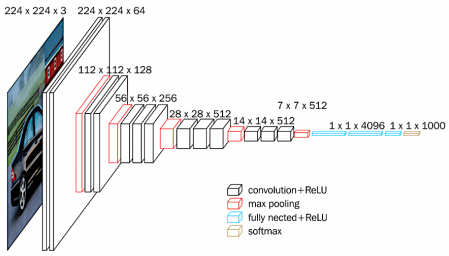
In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3 × 3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

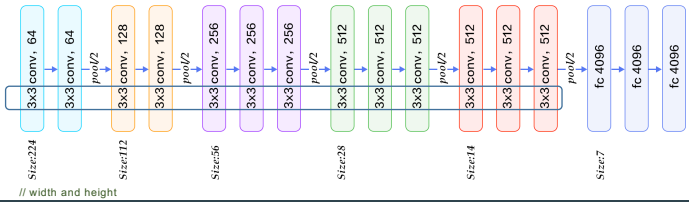
62

Other CNNs

VGG16

- Its main contribution was in showing that the depth of the network (# of layers) is a critical component for good performance.
- Simplified Architecture
  - $Conv = 3\times3\ filter,$ $s = 1, same\ padding$
  - Max-Pool = 2×2 , $s = 1$
- 138M parameters





63

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** Jun 6, 2019

The artificial-intelligence industry is often compared to the oil industry: once mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.

In a new paper, researchers at the University of Massachusetts, Amherst, performed a life cycle assessment for training several common large AI models. They found that the process can emit more than 626,000 pounds of carbon dioxide equivalent—nearly five times the lifetime emissions of the average American car (and that includes manufacture of the car itself).

Common carbon footprint benchmarks

in lbs of CO2 equivalent

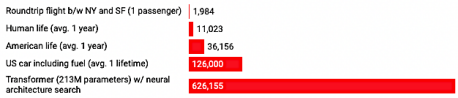


Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

65

Sabour, S., Frosst, N. and Hinton, G.E., 2017. Dynamic routing between capsules. In Advances in neural information processing systems (pp. 3856-3866).

Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

Abstract

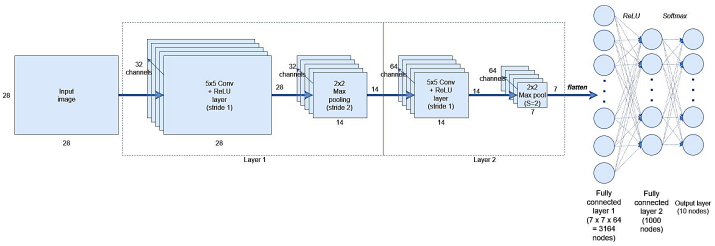
A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. We use the length of the activity vector to represent the probability that the entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions, via transformation matrices, for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. We show that a discrimininatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits. To achieve these results we use an iterative routing-by-agreement mechanism: A lower-level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule.

72

Lab

CNN

- Keras + Tensorflow
- Use Anaconda
- The MNIST database (Modified National Institute of Standards and Technology database)
  - 60,000 training images and 10,000 testing images.
  - The best performance is 0.26 percent error rate.



Questions?

paul@dcs.bbk.ac.uk

