EECS 4422/5323 Computer Vision

Image Understanding 1

Calden Wloka

2 October, 2019

Announcments

- Reminder: Assignment 1 due today
- Reminder: Project proposals due next week

Outline

- Technical Writing
- Introduction to Image Understanding
- Neural Networks
- Convolutional Neural Networks

Important Characteristics of Technical Writing

- Concise - you are trying to communicate maximum information for a given number of words

Important Characteristics of Technical Writing

- Concise - you are trying to communicate maximum information for a given number of words
- Clear - try to avoid ambiguity and logical leaps, appropriately define terminology and variables

Important Characteristics of Technical Writing

- Concise - you are trying to communicate maximum information for a given number of words
- Clear - try to avoid ambiguity and logical leaps, appropriately define terminology and variables
- Supported - factual claims should be supported by your own work or with appropriate external reference

Logical Flow

A technical document is still at heart a story. You, the author, are an expert with the whole picture in your head, but you must recognize that your reader lacks your full perspective, and it is your job to communicate it to them.

- Your document incrementally reveals the pertinent information on the topic in a linear order

Logical Flow

A technical document is still at heart a story. You, the author, are an expert with the whole picture in your head, but you must recognize that your reader lacks your full perspective, and it is your job to communicate it to them.

- Your document incrementally reveals the pertinent information on the topic in a linear order
- It is important not to assume knowledge or understanding of parts which have not yet been presented

Logical Flow

A technical document is still at heart a story. You, the author, are an expert with the whole picture in your head, but you must recognize that your reader lacks your full perspective, and it is your job to communicate it to them.

- Your document incrementally reveals the pertinent information on the topic in a linear order
- It is important not to assume knowledge or understanding of parts which have not yet been presented
- Connections between sections (or look-aheads) can be fostered by cross-references

Organization

A strong standard template for developing a paper or technical report is as follows:

- Motivation and background - put your project in a wider context and explain why it matters

Organization

A strong standard template for developing a paper or technical report is as follows:

- Motivation and background - put your project in a wider context and explain why it matters
- Methodology - explain how you did the work (*e.g.*, models, tools, metrics)

Organization

A strong standard template for developing a paper or technical report is as follows:

- Motivation and background - put your project in a wider context and explain why it matters
- Methodology - explain how you did the work (*e.g.*, models, tools, metrics)
- Results - what are the results of your work and experiments?

Organization

A strong standard template for developing a paper or technical report is as follows:

- Motivation and background - put your project in a wider context and explain why it matters
- Methodology - explain how you did the work (*e.g.*, models, tools, metrics)
- Results - what are the results of your work and experiments?
- Discussion - what can you conclude from the results you obtained?

Organization

A strong standard template for developing a paper or technical report is as follows:

- Motivation and background - put your project in a wider context and explain why it matters
- Methodology - explain how you did the work (*e.g.*, models, tools, metrics)
- Results - what are the results of your work and experiments?
- Discussion - what can you conclude from the results you obtained?

Obviously for your proposals you don't have results to discuss, but the first two points should stand to cover what you *will* do.

Some Final Notes

- Avoid list finishers, *e.g.* "etc.", "and so on"

Some Final Notes

- Avoid list finishers, *e.g.* "etc.", "and so on"
- Follow instructions

Some Final Notes

- Avoid list finishers, *e.g.* “etc.”, “and so on”
- Follow instructions
- Use requirements as a guide

Some Final Notes

- Avoid list finishers, *e.g.* "etc.", "and so on"
- Follow instructions
- Use requirements as a guide
- Number equations, sections, tables, and figures to facilitate cross-referencing and feedback

Some Final Notes

- Avoid list finishers, *e.g.* "etc.", "and so on"
- Follow instructions
- Use requirements as a guide
- Number equations, sections, tables, and figures to facilitate cross-referencing and feedback
- Try to view critical feedback as helpful

Moving Beyond Image Processing

Most of what we have covered so far in this course has been *image processing* - the manipulation of images with the end goal remaining human interpretation and information visualization.

Moving Beyond Image Processing

Most of what we have covered so far in this course has been *image processing* - the manipulation of images with the end goal remaining human interpretation and information visualization.

This topic will shift the emphasis firmly to computer vision, in which the goal is designing algorithms which can independently process images and come to useful conclusions without human input.

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

- Image recognition

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

- Image recognition
- Object localization

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

- Image recognition
- Object localization
- Semantic segmentation

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

- Image recognition
- Object localization
- Semantic segmentation
- Faces
    - Recognition and Identification
    - Emotion Classification

Problem Domains

Image understanding encompasses a very broad range of problem domains. Popular areas of ongoing investigation include:

- Image recognition
- Object localization
- Semantic segmentation
- Faces
    - Recognition and Identification
    - Emotion Classification
- Scene recognition

For all these areas, the dominant approach is deep learning, and convolutional neural networks (CNNs) in particular.

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

1. Neural network and deep learning crash course

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

1. Neural network and deep learning crash course
2. Attention

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

1. Neural network and deep learning crash course
2. Attention
3. Shape and texture

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

1. Neural network and deep learning crash course
2. Attention
3. Shape and texture
4. Feature binding and border ownership

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

1. Neural network and deep learning crash course
2. Attention
3. Shape and texture
4. Feature binding and border ownership
5. Additional topics (depending on previous lectures)

Topic Outline

To explore the topic of image understanding, we are going to take the following schedule:

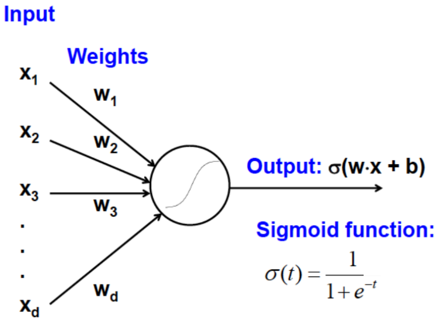
1. Neural network and deep learning crash course
2. Attention
3. Shape and texture
4. Feature binding and border ownership
5. Additional topics (depending on previous lectures)
6. Capsule networks (guest lecture - Nick Frosst)

Deep Learning is Based on Neural Networks

The fundamental model underlying deep learning approaches is the *neural network*, sometimes also called *artificial neural networks* (ANNs).

Neural networks are made up of processing units (*neurons*) which are essentially linear summation units coupled with a non-linear activation function.

One of the earliest formulations comes from Rosenblatt's *Perceptron* model, which used a sigmoid function as the non-linear activation function.

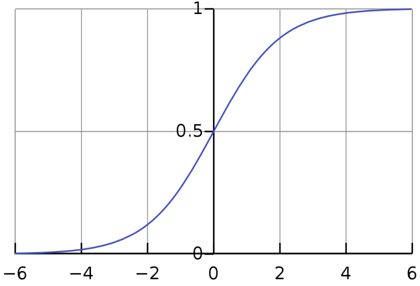


Source: Modified from Richard Wildes' slides from 2017.

The Sigmoid Function

The sigmoid function (also known as the *logistic function*) has a number of attractive properties which motivated its selection:

- Approximately models the firing rates of actual neurons, with a minimum activation (no action potentials) and maximum firing rate
- Smooth and parametric
- Differentiable

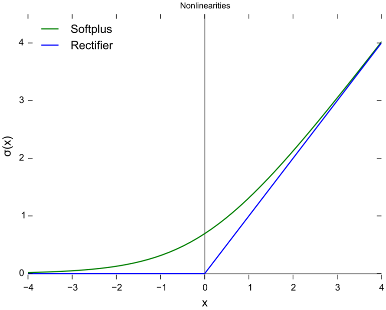


Source: Wikipedia

Rectifier

Most modern neural networks use a different activation function: a *rectifier*.

*Re*ctified *L*inear *U*nits are often referred to by the acronym *ReLU*.

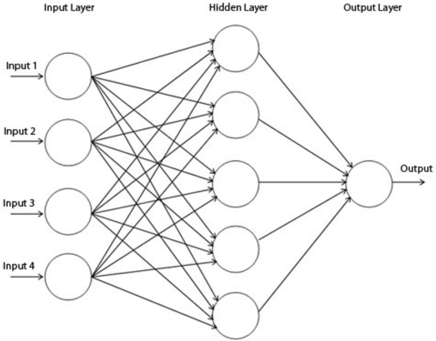


Source: Wikipedia

Neurons are connected to form networks

Any layer which is not either an *input* or *output* layer is known as a *hidden* layer (sometimes also called an *intermediate* layer).

The number of layers of the network is referred to as the network *depth*.



Source: Rosenblatt, 1962

Loss Function

Network weights are learned based on the minimization of an *error signal*, commonly referred to as a *loss function*. The most basic loss function is generated for a given input $\mathbf{x}$ by taking the difference between a true training label, $\mathbf{y}$, and the network prediction $f_{\mathbf{w}}(\mathbf{x})$ for a given set of network weights, $\mathbf{w}$. For a set of N classes, this error can be given as:

$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f_{\mathbf{w},i}(\mathbf{x}))^2$$

Softmax

A very common framework for outputting class labels from a network over N classes is to apply the *softmax function* to the raw network values. Let c_i be the raw score given by the network to category i. The softmax output, $S(c_i)$ is then given by:

$$S(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$$

Softmax

A very common framework for outputting class labels from a network over N classes is to apply the *softmax function* to the raw network values. Let c_i be the raw score given by the network to category i. The softmax output, $S(c_i)$ is then given by:

$$S(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$$

Properties of the softmax function include:

- All categories are assigned a value in the range $(0, 1)$

Softmax

A very common framework for outputting class labels from a network over N classes is to apply the *softmax function* to the raw network values. Let c_i be the raw score given by the network to category i. The softmax output, $S(c_i)$ is then given by:

$$S(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$$

Properties of the softmax function include:

- All categories are assigned a value in the range $(0, 1)$
- The sum of all category labels is 1, making it a valid probability distribution

Softmax

A very common framework for outputting class labels from a network over N classes is to apply the *softmax function* to the raw network values. Let c_i be the raw score given by the network to category i. The softmax output, $S(c_i)$ is then given by:

$$S(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$$

Properties of the softmax function include:

- All categories are assigned a value in the range $(0, 1)$
- The sum of all category labels is 1, making it a valid probability distribution
- High raw scores are heavily emphasized, while moderate or lower scores are suppressed

Softmax

A very common framework for outputting class labels from a network over N classes is to apply the *softmax function* to the raw network values. Let c_i be the raw score given by the network to category i. The softmax output, $S(c_i)$ is then given by:

$$S(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$$

Properties of the softmax function include:

- All categories are assigned a value in the range $(0, 1)$
- The sum of all category labels is 1, making it a valid probability distribution
- High raw scores are heavily emphasized, while moderate or lower scores are suppressed

We can see this in action with a demo.

Cross-entropy loss

Aside from the intuitive ease of interpreting network output as a probability distribution, it also enables the application of a popular loss function: *cross-entropy loss*.

Cross-entropy loss

Aside from the intuitive ease of interpreting network output as a probability distribution, it also enables the application of a popular loss function: *cross-entropy loss*.

Cross-entropy is a method for comparing two probability distributions, p and q. For discrete distributions over N states, this takes the form:

$$H(p,q) = -\sum_{i=1}^{N} p_i \log(q_i)$$

Cross-entropy loss

Aside from the intuitive ease of interpreting network output as a probability distribution, it also enables the application of a popular loss function: *cross-entropy loss*.

Cross-entropy is a method for comparing two probability distributions, p and q. For discrete distributions over N states, this takes the form:

$$H(p, q) = -\sum_{i=1}^{N} p_i \log(q_i)$$

Typically, we take p to be our ground-truth distribution encoded in a *one-hot* fashion, *i.e.* $p_c = 1$ for the actual correct category c, and $p_i = 0$ for $i \neq c$, and q is the output of our network softmax function.

Network Training

Network error can be viewed as a high dimensional manifold over the parameter space formed by the network weights. Provided that $f_{\mathbf{w}}$ is differentiable, gradient descent can be used to search for minima over this manifold via *backpropagation*.

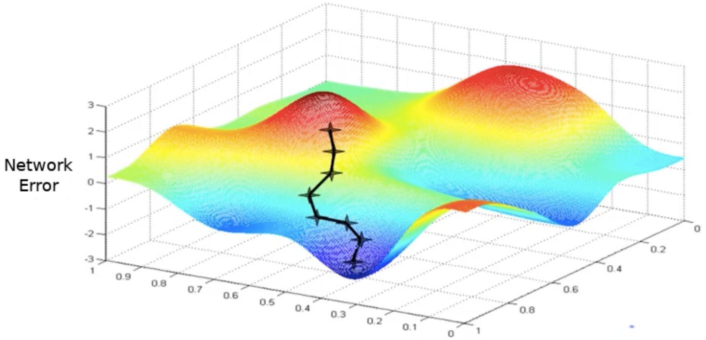


Image source: Original source unknown

What is a CNN?

A convolutional neural network (CNN) is a neural network with a specific connectivity structure based on a series of localized convolutions.

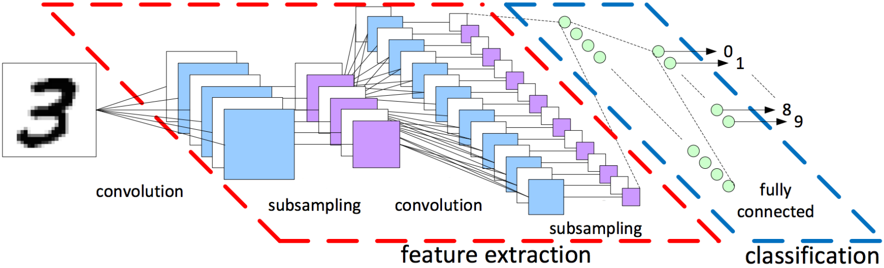


Image source: Original source unknown

Properties of a CNN

- Hierarchical arrangement of stacked filters

Properties of a CNN

- Hierarchical arrangement of stacked filters
- Higher layers extract more global and more abstract features

Properties of a CNN

- Hierarchical arrangement of stacked filters
- Higher layers extract more global and more abstract features
- (Mostly) spatially invariant - a given filter channel utilizes the same kernel over the entire feature map, but sometimes some pixels are skipped to reduce computational load (*stride*))

Common Structure for Convolutional Units

- Our overall architecture is

- But, what is inside each Feature Extractor?

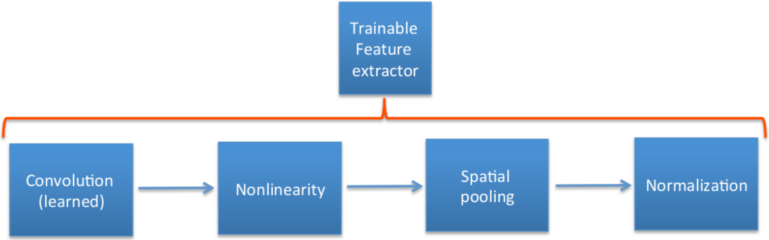
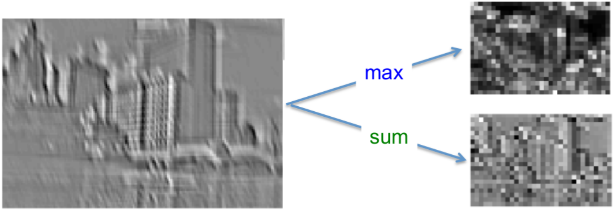


Image source: Adapted from Richard Wildes' lecture slides, 2017

Spatial Pooling

Spatial pooling is typically implemented as an additional filter step to smooth out noise, increase invariance to small image transformations, and to reduce the dimensionality of data to be processed in subsequent layers.

The most commonly used pooling functions are *max* and *sum* (or *average*).

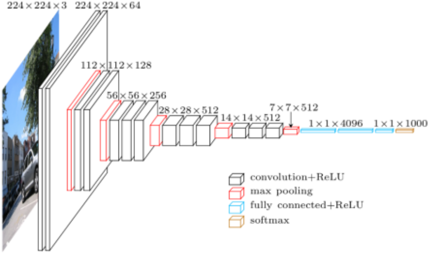


Source: Wildes, 2017

Computational Costs and Training Requirements

As mentioned in prior lectures, a popular CNN is VGG (which actually comes in two major variants, VGG16 and VGG19).

Even the smaller variant of VGG contains 138 million parameters (see Stanford cs231 notes for detailed calculations)



Source: Original image source unknown, VGG by Simonyan & Zisserman, 2014

Computational Costs and Training Requirements

With so many parameters, it becomes very important to train with sufficient data and appropriately designed learning rules to avoid overfitting.

Computational Costs and Training Requirements

With so many parameters, it becomes very important to train with sufficient data and appropriately designed learning rules to avoid overfitting.

This data must be labeled with accurate ground-truth labels, known as *supervised* learning.

Computational Costs and Training Requirements

With so many parameters, it becomes very important to train with sufficient data and appropriately designed learning rules to avoid overfitting.

This data must be labeled with accurate ground-truth labels, known as *supervised* learning.

Given the challenge of ensuring such a large quantity of high quality labeled data is available for training, datasets tend to get reused (risks propagating bias). Ongoing research is devoted to developing *unsupervised* or *semi-supervised* techniques to mitigate this challenge.

Unsupervised Example - Learning Visual Representation by Predicting Sound

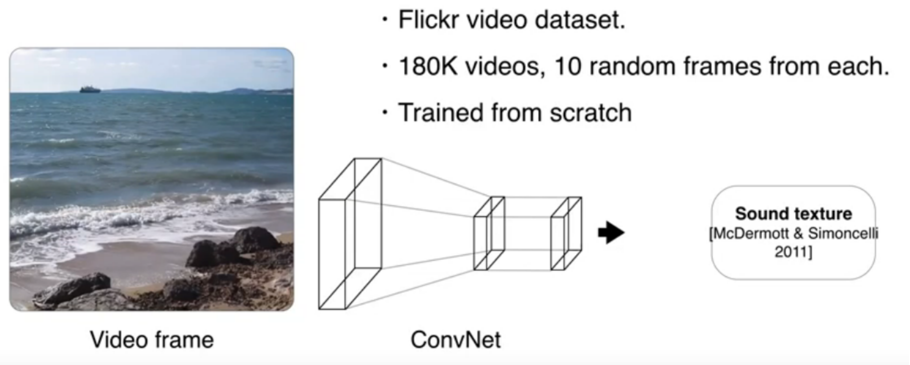


Image source: Screenshot from Torralba's talk, 2018

Transfer Learning

One strategy for mitigating the large training requirements of a deep network is to employ *transfer learning*, whereby you take a network trained on one problem and adapt it to a new problem.

- Learning can be *destructive* or not

Transfer Learning

One strategy for mitigating the large training requirements of a deep network is to employ *transfer learning*, whereby you take a network trained on one problem and adapt it to a new problem.

- Learning can be *destructive* or not
- Non-destructive techniques often freeze the convolution weights and learn a new "readout" network of fully connected layers on top

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?
- *Loss function* - how is error defined?

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?
- *Loss function* - how is error defined?
- *Learning hyperparameters*, such as:

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?
- *Loss function* - how is error defined?
- *Learning hyperparameters*, such as:
  - *Learning rate* - how far do you move over the error manifold with each step?

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?
- *Loss function* - how is error defined?
- *Learning hyperparameters*, such as:
    - *Learning rate* - how far do you move over the error manifold with each step?
    - *Regularization term* - additional term in the error signal to try and smooth it to prevent overfitting

Designing a Network

To set up a network, there are a number of elements which must be specified:

- *Architecture* - how many layers are there, what do they do, and how are they connected?
- *Loss function* - how is error defined?
- *Learning hyperparameters*, such as:
  - *Learning rate* - how far do you move over the error manifold with each step?
  - *Regularization term* - additional term in the error signal to try and smooth it to prevent overfitting
  - *Batch size* - often error is not updated for each input, but rather is updated after a batch of inputs combined into an average error signal

Discussion

- What are some potential risks when using deep networks, or problem domains that are poorly suited to deep learning?

Discussion

- What are some potential risks when using deep networks, or problem domains that are poorly suited to deep learning?
- On the positive side, deep learning can be a very impressive tool (demo)