Neural network programming: image classification

Erik Spence

SciNet HPC Consortium

9 May 2023

Today's code and slides

You can get the slides and code for today's class at the SciNet Education web page.

`https://scinet.courses/1271`

Click on the link for the class, and look under "Lectures", click on "Image Classification".

The best contact address is courses@scinet.utoronto.ca.

Today's class

Today's class will cover the following topics:

- Image classification challenges, data sets.
- AlexNet.
- ResNet.
- Inception.
- Xception.

Please ask questions if something isn't clear.

Image classification

Image classification has been a long-standing area of research in computer vision. As you might imagine, neural networks have been used extensively in the recent past.

- State-of-the-art image classification neural networks can now classify images into one of thousands of classes.
- The standard way to approach these problems is with deep CNNs. They have been quite successful, though they take forever to train.
- This is due to the size of the networks, and the size of the data sets.
- This is one piece of the larger field of object detection: given some sub-photo, classify the image into one of n classes.

Image classification has come a long way in the last decade.

Image classification, data sets

Image classification has been greatly encouraged by the use of competitions, the goal of which is to have the highest classification rate. To this end, many image data sets have been developed.

- CIFAR-10, CIFAR-100, 32 x 32 pixel colour images, with 10 and 100 classes each.
- STL-10, 96 x 96 pixels, colour, 10 classes. Many more unlabelled than labelled.
- SVHN (Street View House Number), colour, 10 classes.
- PASCAL VOC (Visual Object Classes), colour, 20 classes.
- ImageNet, colour, over 20000 classes.
- COCO (Common Objects in Context), colour, hundreds of millions of images, 80 categories, segmentation, captions.

Several of these data sets are being expanded all the time.

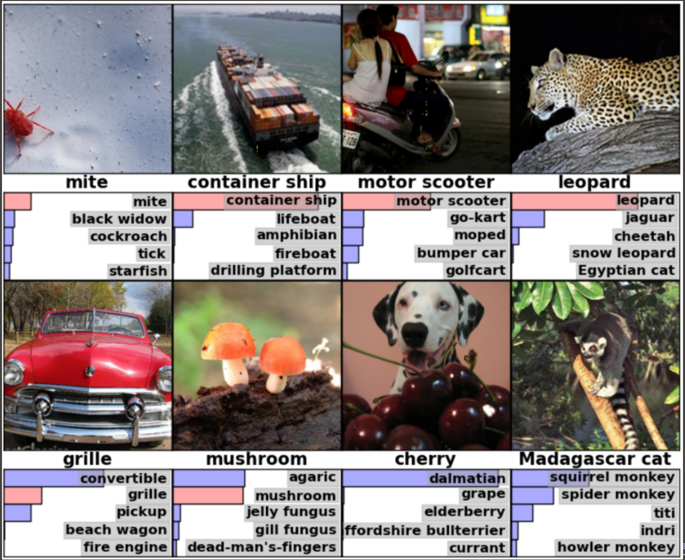
ILSVRC

From 2010 - 2017 the most pre-eminent image classification competition was ILSVRC ("ImageNet").

- Stands for ImageNet Large Scale Visual Recognition Challenge.
- Millions of training images, 1000 categories.
- More-recent competitions also had object-detection challenges, which involved also locating objects within images, and now also within videos.
- The classification competitions allowed the top-5 classifications to be submitted for any given image, along with the associated bounding boxes for each object.
- In 2017, 29 of 38 teams had greater than 95% accuracy. This is probably why the challenge does not appear to be running anymore.

This competition saw significant innovations in neural network architectures.

ILSVRC, top-5 example



A note on human accuracy

How well the classifying neural networks did, relative to humans, became a matter of debate in the last few years of the competition.

- "Are the networks super-human?" was a question that couldn't be answered.
- The ImageNet people themselves looked into this matter, and estimated the human error rate to be about 5% on the classification problem.
- Part of this is due to ambiguity in what the image is supposed to be showing.
- Part of this is due to difficulty in distinguishing small details in the images (dog breeds).

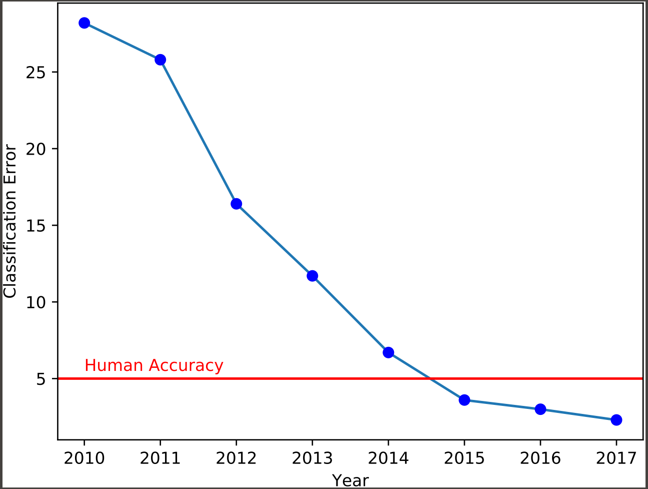
Because neural networks now exceed human accuracy, this competition has lost the interest of the major players.

ILSVRC, continued

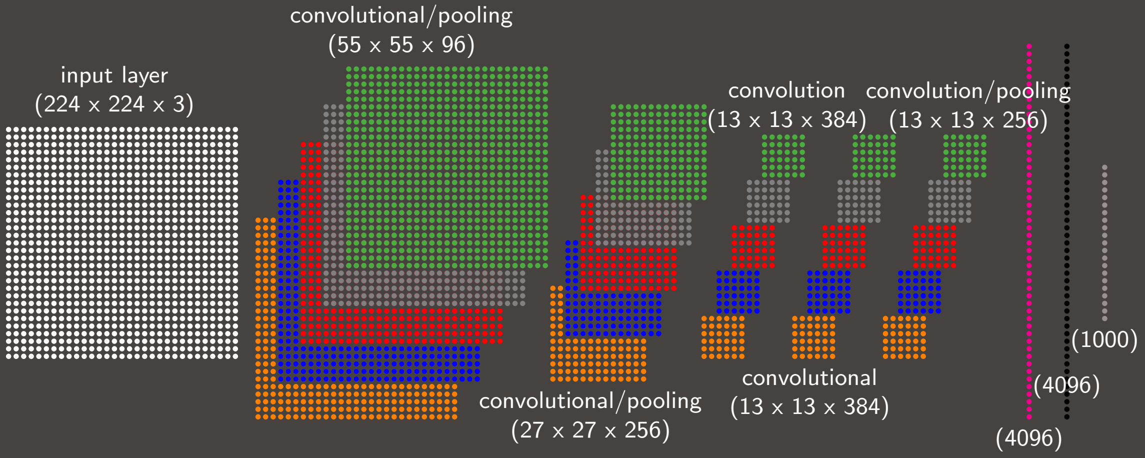
ILSVRC winners introduced new ideas to image classification.

- 2012, AlexNet: first network to use successive convolutional layers in the competition.
- 2014, GoogLeNet: introduced the "Inception Module".
- 2015, ResNet: introduced the "ResNet Module".

Let's review these networks, and the innovations they introduced.



AlexNet (2012)



AlexNet, continued

AlexNet was a ground-breaking neural network.

- Introduced in 2012 by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever, from the SuperVision group of UofT (the group that invented dropout, relu, ...).
- Winner of the ImageNet Large Scale Visual Recognition Challenge in 2012.
- It destroyed the competition, beating the next-best competitor by 10%.
- It was the first neural network to use multiple convolutional layers in succession in an image classification competition.
- The following year, almost all entries in the ImageNet competition contained multiple convolutional layers.
- Notice that the last few convolutional networks aren't shrinking. These networks are padded with zeros to prevent the shrinking of the output.

This was a seminal step in image classification neural networks.

Problems with deep networks

Why do very deep neural networks tend to perform worse as they get deeper?

- In theory at least, a network with $n + 1$ layers should perform *no worse* than a network with n layers.
- In fact, according to the hand waving I've shared with you, more layers should be able to learn more-complex patterns, and so improve the accuracy.
- However, in practice, this is not always the case. Deep neural networks have a habit of failing to train, due to the "vanishing gradient" problem.
- The vanishing gradient problem:
  - layers which are close to the cost function train quickly, due to large gradients,
  - layers which are farther away from the cost function have small gradients, and train slowly, if at all.
- This is a general feature of the neural networks which we've looked at thus far.

How do we get past this problem?

Problems with deep networks, continued

There are several things that can be done to help the network train when it is deep.

- Don't use sigmoid! Both tanh and relu have stronger gradients, which help mollify the vanishing gradient problem.
- Use better random initialization. It turns out that using a Gaussian with a standard deviation of 1 results in a distribution which is way too wide. This can result in "neuron saturation", whereby the gradients of the activation functions go to zero (at least for sigmoid and tanh).
- Instead, if a neuron has n inputs, then the standard deviation of the initialization Gaussian should be $1/\sqrt{n}$. This will result in a distribution sharply peaked around 0.
- Change the architecture of the network, to help avoid these problems.

Let's examine that last point in more detail.

ResNet (2015)

ResNet was developed to address the problem of poorly training deep networks.

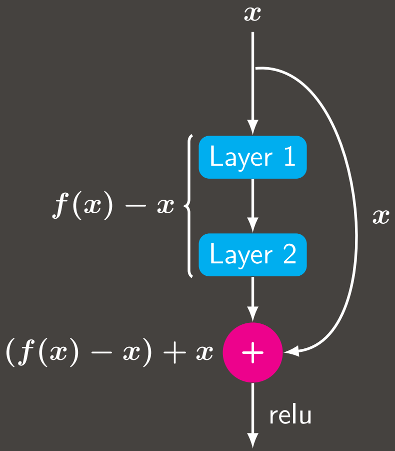
- The authors of ResNet proposed a hypothesis: the reason $n+1$ layers do not necessarily perform as well as n layers is because having a neural network layer learn a direct mapping ($f(x) \rightarrow x$), is difficult.
- They proposed a solution: instead of trying to have a layer learn some particular mapping, $f(x)$, have the layer learn the difference between the desired mapping and the input: $g(x) = f(x) - x$, *i.e.* the residual.
- Assuming this is a small thing to learn, we can then add the original input to the layer, x, to the output of the layer, $f(x) - x$, to get what we wanted in the first place, $f(x)$.
- If the differences are small, they will hopefully be easier to train.

This was a new idea, and it turns out to work quite well. This technique is now commonly used in all-manner of neural networks.

ResNet block

ResNet is built out of 'ResNet blocks'.

- Each block is made of a series of layers.
- The input data enters the series of layers as usual, but also bypasses the layers using a shortcut. This input data is then added to the output of the layers.
- The addition is element-wise. If the input and output are of different sizes, zero-padding is used to get matching dimensions.
- The activation function is applied after the addition.
- This process seems to work better when multiple layers are bypassed.



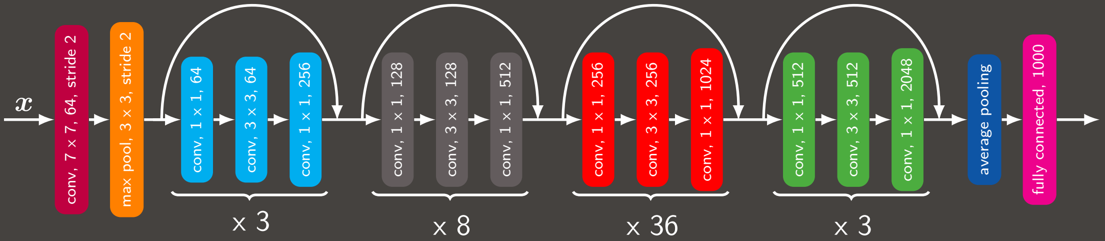
ResNet, continued

ResNet blocks make training of deep networks easier.

- If you add the ResNet block to an existing neural network, with n layers, one would expect that the result would at least be no worse than without the additional block.
- This is because, in that case, all that would need to be learned is a residual of $\mathbf{0}$, i.e. $g(x) = f(x) - x = \mathbf{0}$. This shouldn't be that hard.
- This actually works remarkably well, completely suppressing the vanishing gradient problem.
- Why? Because the shortcuts to the layer inputs allows the derivatives' to bypass the layers (or so goes the hand waving).
- Neural networks with over 1000 layers have been built, and trained, by using this construction.

Needless to say, this was a big step forward.

ResNet 152 (2015)



Rather than bypass just 2 layers, ResNet 152 is built out of blocks which bypass 3 layers.

The winner of ILSVRC 2015. 152 layers!

Inception

The authors of Inception asked a different question: how can we make bigger neural networks without increasing the computational costs of training?

- This group introduced what is known as the "Inception module". It's based on two observations.
- The first relates to the choice of hyperparameters pertaining to the output of a given convolutional layer (or even in the input layer, for that matter).
- In what I've shown you so far, we've had to input the kernel size by hand ((5 x 5) or (3 x 3) or whatever).
- What would happen if, instead of us deciding what size of kernel is the best one, the network itself got to decide which size of kernel to use?
- Or for that matter, why not let the neural network decide whether to do a convolutional layer versus a pooling layer?

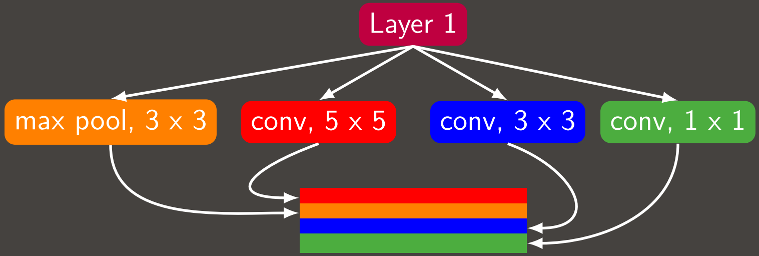
Inception, continued

This sounds pretty slick. How would we do that?

- One approach would be to calculate all the convolutional kernel, and max-pooling kernel, sizes you might be interested in, in parallel.
- These results could then be concatenated into a single output. The following layer in the neural network could then decide which data is most useful.
- The problem with this approach is that it doesn't reduce the computational cost of our training. Quite the opposite, in fact. Since the amount of computation is going up quadratically.

First let's see what this looks like.

Inception, first idea



The outputs of the layers are bundled into a single input for the next layer.

This only increases the computation time. How do we deal with that?

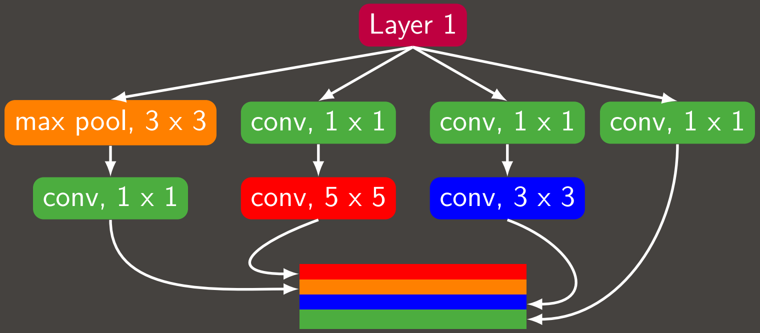
Inception, continued more

How do we overcome the increase in computation?

- The approach adopted by the authors was to use 1 x 1 convolutional filters to reduce the dimensionality of the data coming into any given layer.
- So if, say, the output of your convolutional layer is (24, 24, 20), you can run a (1 x 1) convolutional filter on the output, with a different number of feature maps, to make the output, say, (24, 24, 10). This halves the amount of computation needed in the next layer.
- Notice that (1 x 1) filters do not do any spatial (2D) analysis of the incoming data, they only analyze the data across feature maps.
- The first Inception network was called GoogLeNet, presumably named after LeNet-5 (1998), which was an early image classification neural network.

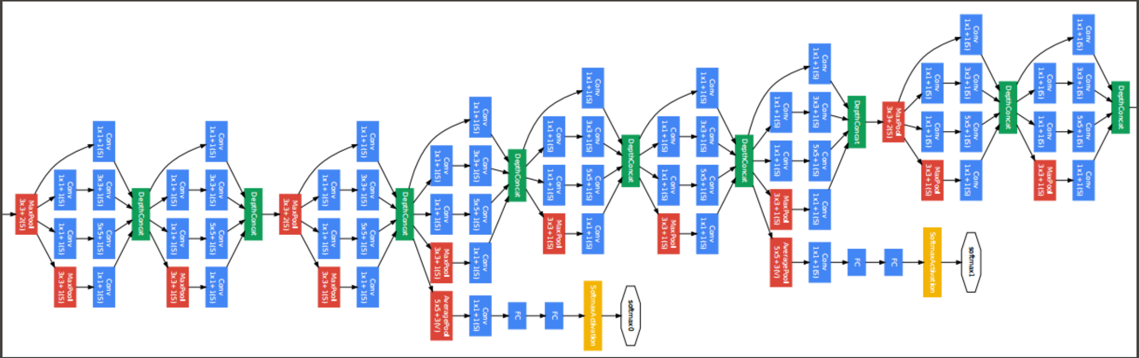
GoogLeNet won the ILSVRC 2014 competition.

Inception, second idea



1 x 1 convolutional layers are used to reduce the dimensionality of the input, making the computation more manageable.

GoogLeNet (2014)



GoogLeNet won the ILSVRC 2014 competition. 22 layers, but fewer trainable parameters than AlexNet.

Inception, continued lots more

But it doesn't end there. The authors came up with even more innovations.

- Version 2 of the network refactored larger convolutions into successive smaller convolutions (2 3 x 3 convolutions instead of a single 5 x 5), resulting in fewer parameters to train.
- Version 3 added Batch Normalization to the fully connected layers.
- Version 4 has added ResNet-type short cuts within each Inception Module.

By now, Inception networks exceed human performance on ImageNet data.

Xception (2016)

The Xception network is so named because it takes the Inception network to the "extreme".

The hypothesis being tested by the author was the following: cross-channel correlations and spatial correlations are sufficiently decoupled that it is better to not map them together.

- The typical convolutional layer looks for correlations across space and feature maps.
- If your convolutional layer has a (3 x 3) kernel, and there are 20 incoming feature maps, then the number of weights for a single feature map in the current layer is (3 x 3) x 20. (We are looking for correlations across space and channels.)
- We already started to decouple space and cross-channel (feature maps) in the Inception approach.
- We did this by reducing the number of channels by applying a (1 x 1) convolution across all channels.

The Xception network completely separates the search for spatial and cross-channel correlations.

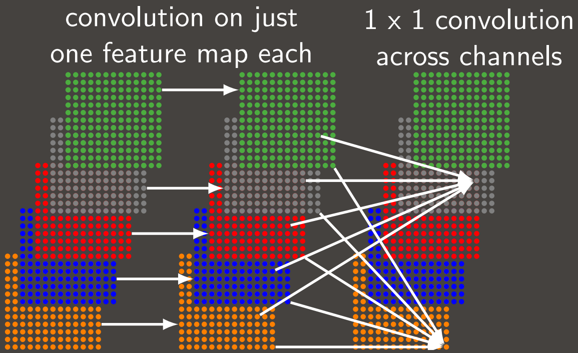
Xception, continued

Let's take this to the extreme.

- The Xception approach is to run a single convolutional feature map on each incoming channel separately, and then to perform a (1 x 1) convolution across all channels.
- This is essentially the same as what is called a "depthwise separable convolution", followed by a pointwise convolution (1 x 1 convolution across all channels).
- You can think of this as first searching for 2D correlations in space, followed by looking for 1D correlations in depth.
- This is easier to train than a full 3D correlation search, and there are far fewer trainable parameters.
- This network architecture outperforms Inception v3, and does significantly better on classification data sets with tens of thousands of categories.

The author of Xception is also the author of Keras (Francois Chollet).

Xception module


convolution on just one feature map each
1 x 1 convolution across channels

Xception networks are built out of this module, combined with max-pooling layers.

Xception, continued more

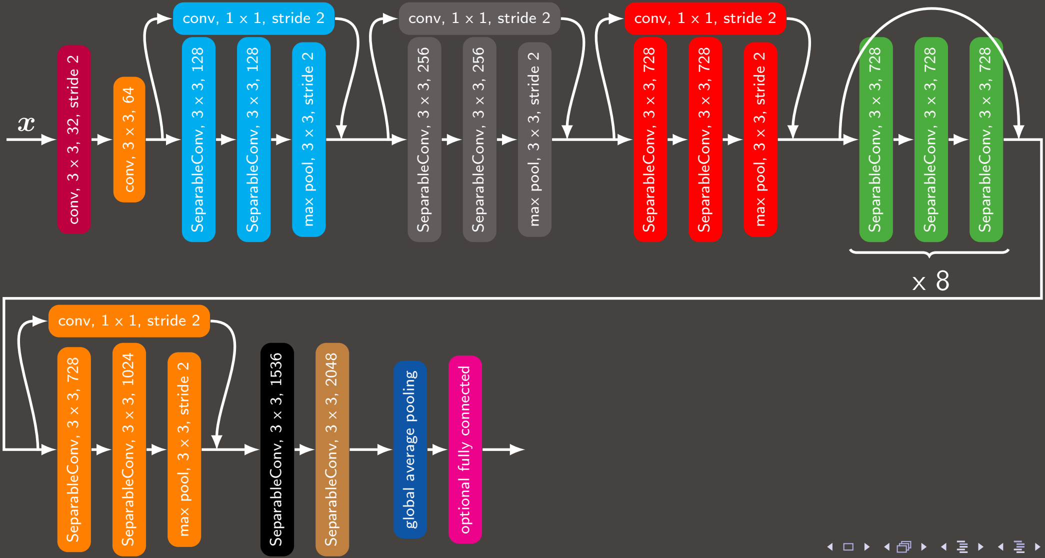
How good is the improvement in the number of trainable parameters?

Suppose we have 20 channels as the input to a given convolutional layer, and we want to use a (3 x 3) filter, and output 40 feature maps (channels).

- The original way to do the convolution would result in ((3 x 3) x 20 + 1) x 40 = 7240 free parameters.
- The Xception way is to
  - first do the spatial correlation, but only on one feature map at a time. This gives ((3 x 3) + 1) x 20 = 200.
  - then do the correlations across channels. This gives ((1 x 1) x 20 + 1) x 40 = 840.
- For a total of 1040 parameters.

Almost a factor of 7 improvement. And because the two sets of correlations are separated from each other, it's easier for the network to learn.

Xception network (2016)



x
conv, 3 x 3, 32, stride 2
conv, 3 x 3, 64
conv, 1 x 1, stride 2
SeparableConv, 3 x 3, 128
SeparableConv, 3 x 3, 128
max pool, 3 x 3, stride 2
conv, 1 x 1, stride 2
SeparableConv, 3 x 3, 256
SeparableConv, 3 x 3, 256
max pool, 3 x 3, stride 2
conv, 1 x 1, stride 2
SeparableConv, 3 x 3, 728
SeparableConv, 3 x 3, 728
max pool, 3 x 3, stride 2
SeparableConv, 3 x 3, 728
SeparableConv, 3 x 3, 728
SeparableConv, 3 x 3, 728
x 8
conv, 1 x 1, stride 2
SeparableConv, 3 x 3, 728
SeparableConv, 3 x 3, 1024
max pool, 3 x 3, stride 2
SeparableConv, 3 x 3, 1536
SeparableConv, 3 x 3, 2048
global average pooling
optional fully connected

Image classification, Keras models

Keras ships with six pre-trained models (layers and weights), built into it.

- VGG16, VGG19 (2014)
- ResNet50, ResNet101, ResNet152
- Inception v3
- Xception
- MobileNet

These networks can be accessed through the keras.applications subpackage.

The VGG networks follow the same basic layout as AlexNet, which preceded all of the other above networks.

Linky goodness

ILSVRC:

- `http://www.image-net.org/challenges/LSVRC`

Networks:

- AlexNet: `http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`
- ResNet: `https://arxiv.org/abs/1512.03385`
- GoogLeNet: `https://arxiv.org/abs/1409.4842`
- Xception: `https://arxiv.org/abs/1610.02357`