Slide #2

Let's begin with a quick overview of why we are interested in convolutional neural networks. In earlier lectures, we talked about image data, where each image consisted of a two-dimensional grid of pixels. Depending on whether we are handling black-and-white or color images, each pixel location might be associated with either one or multiple numerical values. Before learning about convolutional neural networks, we simply discarded the spatial structure of each image by flattening them into one-dimensional vectors. These vectors were then fed through a fully-connected multilayer perceptron (MLP) consisting of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Because these networks are invariant to the order of the features, we could get similar results regardless of whether we preserve the spatial structure of the pixels or if we permute the columns of our design matrix before fitting the hyperparameters. Ideally, we would implement our prior knowledge that nearby pixels are typically related to each other. We would then use this information to build efficient models for learning from the image data.

Slide #3

With this information in mind, I will now start our "tour" of modern CNN architectures. Each of the networks shown here were briefly a dominant architecture for supervised learning in computer vision. These models include: AlexNet, the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge; VGG, which makes use of a number of repeating blocks of elements; NiN, which convolves whole neural networks patch-wise over inputs; GoogLeNet, which uses networks with parallel concatenations; ResNet, which remains the most popular off-the-shelf architecture in computer vision; and DenseNet, which is computationally expensive but has set some recent benchmarks in computer vision. We went over the first two architectures during last lecture on Monday. I am going over the next two architectures during this lecture today. We will go over the final two architectures during next lecture on Monday. While the ideas behind these neural networks are quite simple (basically, we just stack together a bunch of layers), it's important to note that performance can vary wildly between different architectures and different hyperparameter choices.

Slide #4

Before diving into the details of NiN or GoogLeNet, which are the topic of today's lecture, I was curious about (1) the timeline and (2) the relative importance of these modern CNN architectures. The two most important architectures (as determined by their number of citations) are AlexNet and ResNet, with AlexNet being the first large-scale architecture deployed in computer vision and ResNet being the most popular off-the-shelf architecture in computer vision today.

Slide #5

Let's begin with NiN.

Slide #6

LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolution and pooling layers and then post-process the representations via fully-connected layers. Alternatively, one could imagine using fully-connected layers earlier in the process. However, a careless use of dense layers might give up the spatial structure of the representation entirely. Network in network (NiN) blocks offer an alternative to this and they were proposed based on a very simple insight: to use a MLP on the channels for each pixel separately. Recall that the inputs and outputs of convolutional layers consist of four-dimensional tensors with axes corresponding to the example, channel, image height, and image width. Also recall that the inputs and outputs of fully-connected layers are typically two-dimensional tensors corresponding to the example and feature. The idea behind NiN is to apply a fully-connected layer at each pixel location. There's two ways to think about this layer. We can either think of this (1) as a 1x1 convolutional layer or (2) as a fully-connected layer acting independently on each pixel location. The figure shown here illustrates the main structural differences between VGG and NiN, as well as their blocks. The NiN block consists of one convolutional layer followed by two 1x1 convolutional layers that act as per-pixel fully-connected layers. The convolution window shape of the first layer is typically set by the user. The subsequent window shapes are fixed to 1x1. The original NiN network shown here was proposed shortly after AlexNet. NiN uses convolutional layers with window shapes of 11x11, 5x5, and 3x3. These design choices are the same as in AlexNet. Each NiN block is followed by a maximum pooling layer with a stride of 2 and a window shape of 3x3. One significant difference between NiN and AlexNet is that NiN avoids fully-connected layers altogether. One advantage of NiN's design is that it significantly reduces the number of required model parameters. However, in practice, this design sometimes results in a longer training time.

Slide #7

I was also curious as to what astrophysical applications still use NiN. I'm showing here the five most recent refereed astrophysical papers from ADS.

Slide #8

Let's end this section with a quick summary of NiN: (1) NiN uses blocks consisting of a convolutional layer and multiple 1x1 convolutional layers; (2) NiN removes the fully-connected layers and replaces them with global average pooling layers after reducing the number of channels to the desired number of outputs; (3) removing the fully-connected layers reduces overfitting and results in NiN having dramatically fewer parameters; and (4) the NiN design influenced many subsequent CNN designs.

Slide #9

Let’s move onto GoogLeNet.

Slide #10

GoogLeNet won the ImageNet Challenge in 2014 and proposed a structure that combined the strengths of NiN and paradigms of repeated blocks. One focus of the original paper was to address the question of which sized convolution kernels are best. This is a relevant question since the previous networks that we talked about employed seemingly random choices as small as 1x1 and as large as 11x11. One key insight in this paper was that sometimes it can be advantageous to employ a combination of variously-sized kernels. The basic convolutional block in GoogLeNet is called an Inception block. The inception block consists of four parallel paths. The first three paths use convolutional layers with window sizes of 1x1, 3x3, and 5x5 to extract information from different spatial sizes. The middle two paths perform an additional 1x1 convolution on the input to reduce the number of channels, reducing the model's complexity. Note that this is different from the 1x1 convolutions performed in NiN, which enhance model complexity by allowing interactions of cross channel information. The fourth path uses a 3x3 maximum pooling layer, followed by a 1x1 convolutional layer to reduce the number of channels. The four paths all use appropriate padding to give the input and output the same height and width. Finally, the outputs along each path are concatenated along the channel dimension and comprise the block's output. GoogLeNet uses a stack of a total of 9 inception blocks and global average pooling to generate its estimates as shown in this figure. Maximum pooling between inception blocks reduces the dimensionality. The first module is similar to AlexNet and LeNet. The stack of blocks is inherited from VGG and the global average pooling avoids a stack of fully-connected layers at the end. To gain some intuition for why this network works so well, let's consider why we are combining filters. When we combine filters, we are effectively exploring the image in a variety of filter sizes. This means that details at different extents can be recognized efficiently by filters of different sizes, which is good for us.

Slide #11

I was again curious as to what astrophysical applications still use GoogLeNet. I’m again showing the five most recent refereed astrophysical papers from ADS.

Slide #12

Let's end this section with a quick summary of GoogLeNet: (1) the Inception block is equivalent to a subnetwork with four paths, it extracts information in parallel through convolutional layers of different window shapes and maximum pooling layers, 1x1 convolutions reduce channel dimensionality on a per-pixel level, maximum pooling reduces the resolution; (2) GoogLeNet connects multiple well-designed Inception blocks with other layers in series; and (3) GoogLeNet was one of the most efficient models on ImageNet, providing similar test accuracy to other networks while requiring lower computational complexity.

Slide #13

Some of you may have noticed that there was one recent paper on ADS that referenced both NiN and GoogLeNet. I will now conclude today's lecture by talking about this paper, so that we can get a sense of how these architectures are actually implemented with real data.

Slide #14

This paper is titled "Extracting Photometric Redshift from Galaxy Flux and Image Data using Neural Networks in the CSST Survey". For those unfamiliar, CSST is the Chinese Space Station Telescope, which is a planned space telescope currently under development that will have a 2 meter primary mirror and a 2.5 gigapixel camera. It's planned for lunch in 2024.

Slide #15

Since the CSST has not launched, this paper uses simulated images based on HST observations in the COSMOS field. This figure shows the galaxy redshift distribution of the sample selected from the COSMOS catalog that will be used in the neural networks.

Slide #16

Here's what some of those simulated images look like. CSST has seven photometric filters, with each column corresponding to a different filter. This figure illustrates that some sources, particularly at high redshifts, are dominated by background noise, which may indicate that neural networks are required to extract any useful information from these sources.

Slide #17

Now that we understand their simulated data, let's look at the neural network that they use. Observing a galaxy with CSST would generate seven discrete data points in the seven CSST photometric bands. This paper initially adopts the MLP to predict photometric redshifts from these data. The MLP is the simplest form of neural networks, consisting of an input layer, several hidden layers and an output layer. Every layer has several units or perceptrons, and the hidden layers can learn the internal relationship between input data points. Layers are connected by weights and biases, which can be initialized by random distributions and optimized in the training process. The input of their MLP has 20 values, which are the seven fluxes, seven errors, and six colors. The output of their MLP has 1 value, which is the photometric redshift. The details of the architecture of the MLP are shown in the blue dash-dotted box. This paper then adopts their own CNN to predict photometric redshifts. They make use of the inception blocks from GoogLeNet to extract image features in parallel. The inception block has been widely used for predicting photometric redshift from galaxy images. After the inception blocks, they make use of the global average pooling from NiN to vectorize their image features. The input of their CNN has [32, 32, 7] images. The output of their CNN is the photometric redshift. The details of the architecture of the CNN are shown in the black

dashed box.Since the CNN can extract photometric redshifts from the galaxy image using primarily morphology information, they decide to combine their MLP and CNN into a hybrid network. This means that they can determine photometric redshifts using both the galaxy flux data and the image data. Their fully connected layers are concatenated and followed by a series of fully connected layers. The details of the combined MLP and CNN hybrid network are shown here.

Slide #18

Let’s take a look at their results. These figures are very difficult to parse through at first glance, so I’ll just tell you the takeaway. The MLP using flux data can provide a lower scatter in the predicted photometric redshifts compared to the CNN, however, the CNN using image data can provide a smaller outlier fraction compared to MLP. The hybrid network can improve the results further by lowering both the scatter and outlier fraction.

Slide #19

Let's end this section with a quick summary of this paper: (1) they try to extract photometric information from galaxy flux and image data expected to be obtained by the CSST using neural networks; (2) for their data, they generate mock galaxy images; (3) for their neural networks, they construct an MLP, a CNN, and a hybrid network of the two; and (4) they find that their networks effectively and properly extract photometric redshifts from the simulated CSST data.