

Let's ride to another popular stop on the neural network map.

Recurrent neural networks (such as LSTMs) loop back on themselves repeatedly —

— addressing problems with the temporal element —

— such as speech recognition.

Others, like auto-encoders, helped make sense of unsupervised data —

— reducing the dimensionality of overflowing big data.

Here we are. All out for convolutional neural networks!

CNNs

Aha! I've heard of these.

They get a lotta press.

Well, isn't that true of any neural network that handles images?

Sure! But CNNs offer a unique method for parsing and obtaining all that image-based data.

(Including any data type that can be represented as images.)

Even a low resolution image like this contains a wealth of information. Every one of those 1024 pixels is a separate input — three, in fact, if you count the red, green, and blue channels.

It's Uncle Rufus!

Let's start by building a map of cat-related features in our source image.

CATCH!

What's this?

A filter. It starts out as a matrix of random weights, but the algorithm tunes it up over time.

A "convolution" involves moving this filter across the entire image by of variable interval known as a "stride."

Oh! It's multiplying the source data by that matrix!

The result is called a feature map!

These feature maps go through pooling to further reduce their computational size.

Ooh, so you can keep stacking them to hunt down more features!

CONVOLUTION

ACTIVATION

POOLING

INPUT

In their earliest stages, filters may detect no more than edges and orientation...\*

But with every subsequent layer, composite features start to emerge from the noise.

See? It me!

CNNs aren't fully connected like our first example. But the final output layer is; connecting to all the neurons on the previous layer.

CONVOLUTION

ACTIVATION

CLASSIFICATION

POOLING

OUTPUT

\*FUN FACT: your brain's visual cortex initiates detection this way too... in fact, it was an early inspiration for CNNs.