Tomer Eldor Follow
Passionate about data science & AI for social good and making great user-focused products. A Jazz composer living every 4 months in another country.
Mar 9 · 8 min read

Capsule Neural Networks: The Next Neural Networks? Part 1: CNNs and their problems.

Convolutional ('regular') Neural Networks are the latest hype in machine learning, but they have their flaws. Capsule Neural Networks are the recent development from Hinton which help us solve some of these issues.

> *This is part of a **series** of posts built to teach about Capsule Neural Networks. This first post will explain about "normal" (convolutional) Neural Networks and their problems.*

Neural Networks may be the hottest field in Machine Learning. In recent years, there were many new developments improving neural networks and building making them more accessible. However, they were mostly incremental, such as adding more layers or slightly improving the activation function, but did not introduce a *new* type of architecture or topic.

Geoffery Hinton is one of the founding fathers of many highly utilized deep learning algorithms including many developments to Neural Networks—no wonder, for having Neurosciences and Artificial Intelligence background.

At late October 2017, Geoffrey Hinton, Sara Sabour, and Nicholas Frosst Published a research paper under Google Brain named "Dynamic Routing Between Capsules", introducing a true innovation to Neural Networks. This is exciting, since such development has been long awaited for, will likely spur much more research and progress around it, and is supposed to make neural networks *even better than they are now.*

. . .

The Baseline: Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are extremely flexible machine learning models which were originally inspired by principles from how our brains are theorized to work.

> *Neural Networks utilize layers of "neurons" to process raw data into patterns and objects.*

The primary building blocks of a Neural Network is a "Convolutional" layer (hence the name). What does it do? It takes raw information from a previous layer, makes sense of patterns in it, and send it onward to the next layer to make sense of a larger picture.

If you are new to neural networks and want to understand it, I recommend:

1. Watching the animated videos by 3Blue1Brown.
2. For a more detailed textual/visual guide, you can check out this beginner's blogpost
3. If you can deal with some more math and greater details, you can read instead this guide from CS231 at Stanford.

In case you didn't do any of the above, and plan to continue, here is a hand-wavy brief overview.

The Intuition Behind Convolutional Neural Networks

Let's start from the beginning.
The Neural Net receives raw input data. Let's say it's a doodle of a dog. When you see a dog, you brain automatically detects it's a dog. But to the computer, the image is really just an array of numbers representing the colors intensity in the colors channels. Let's say it's just a Black&White doodle, so we can represent it with one array where each cell represents the brightness of the pixel from black to white.

Example dog face for a Neural Network to recognize. Source: The Sun, image: lovable dog rescue

What is our goal? Our goal is for the net to *make visual sense* of what is in the picture (which is for it just a sequence of numbers). One way to do it is bottom-up: start by looking at small groups of pixels, make sense of them (like small lines and curves: the curve of the dog's ear, the small circle of its pupil, the small line in its leg), then build those up to bigger groups of these lines and curves (like: ear, nose, snout, eye), make sense of bigger groups of these shapes (like: face, leg, tail), and finally make sense of the entire picture of the dog.

It does so with many layers transferring information in a sequence from one to another.
If this was new to you, see my summary of Convolutional Networks Structure here: **Understanding Convolutional Neural Networks.**

In case you didn't read it but it's still new to you, below is an even shorter summary of that summary:

. . .

Understanding Convolutional Neural Networks.

1. **Convolutional Layers**. The first convolutional layer maps the image space to a lower space—summarizing what's happening in each group of, say 5x5 pixels—is it a vertical line? horizontal line? curve of what shape? *This happens with element wise multiplication and then summation of all the values in the filter with the original filter value and summing up to a single number.*

2. This leads to the ***Neuron, or*** convolutional ***filters***. *Each Filter / Neuron is designed to react to one specific form (a vertical line? a horizontal line? etc…).* The groups of pixels from layer 1 reach these neurons, and lights up the neurons that match its structure according to how much this slice is similar to what the neuron looks for.

3. **Activation (usually "ReLU") Layers**—After each convolutional layer, we apply a nonlinear layer (or activation layer), which introduces non-linearity to the system, enabling it to discover also nonlinear relations in the data. ReLU is a very simple one: making any negative input to 0, or if it's positive—keeping it the same. *ReLU(x) = max(0,x).*

4. **Pooling Layers**. This allows to reduce "unnecessary" information, summarize what we know about a region, and continue to refine information. For example, this might be "MaxPooling" where the computer will just take the highest value of the passed this patch —so that the computer knows "around these 5x5 pixels, the most dominant value is 255. I don't know exactly in which pixel but the exact location isn't as important as that it's around there. → ***Notice: This is not good. We loose information here. Capsule Networks don't have this operation here, which is an improvement.***

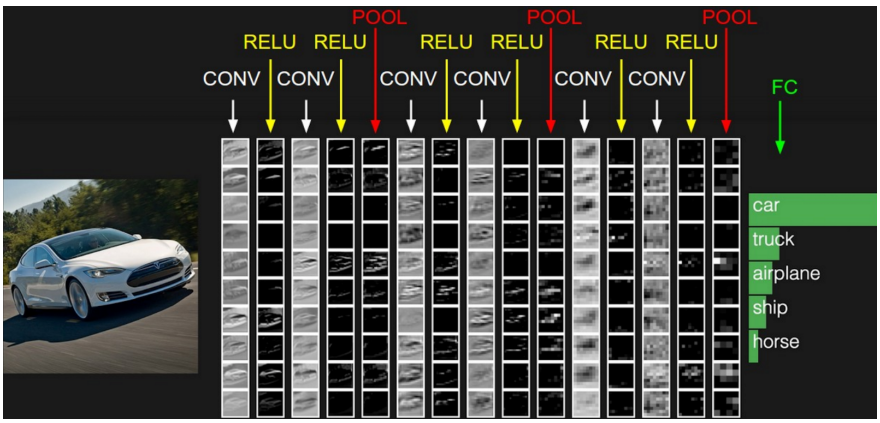
5. **Dropout Layers.** This layer "drops out" a random set of activations in that layer by setting them to zero. This makes the network more robust (kind of like you eating dirt builds up your immunity system, the network is more immune to small changes) and reduces overfitting. This is only used when training the network.

6. **Last Fully Connected Layer.** For a classification problem, we want each final neuron represents the final class. It looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class.

7. **SoftMax—**This layer is sometimes added as a another way to represent the outputs per classes that we can later pass on in a loss function. Softmax represents the distribution of probabilities to the various categories.

Usually, there are more layers which provide nonlinearities and preservation of dimensions (like padding with 0's around the edges) that help to improve the robustness of the network and control overfitting. But these are the basics you need to understand what comes after.

Now, importantly, these layers are connected only SEQUENTIALLY. This is in contrast to the structure of capsule networks.



Neural Network Structure, From a Google article be Szegedy, Toshev & Erhan presenting Neural Networks

. . .

What is The Problem With Convolutional Neural Networks?

If this interests you, watch Hinton's lecture explaining exactly *what it wrong with them. Below you'll get a couple of key points that are improved by Capsule Networks.*

Hinton says that they have too few levels of substructures (nets are composed from layers composed from neurons, that's it); and that we need to group the neurons in each layer into "capsules", like mini-columns, that do a lot of internal computations, and *then* output a summary result.

Problem #1: Pooling looses information

CNN use “pooling” or equivalent methods to “summarize” what's going on in the smaller regions and make sense of larger and larger chunks of the image. This was a solution that made CNNs work well, but it *looses valuable information.*

Capsule networks will compute a ***pose*** (transnational and rotational) relationship between smaller features to make up a larger feature.
This loss of information leads to loss of spatial information.

Problem #2: CNNs don't account for the spatial relations between the parts of the image. Therefore, they also are too sensitive to orientation.

> *Subsampling (and pooling) loses the precise spatial relationships between higher-level parts like a nose and a mouth. The precise spatial relationships are needed for identity recognition.*

(Hinton, 2012, in his lecture).

CNNs don't account for spatial relationships between the underlying objects. By having these flat layers of neurons that light up according to which objects they've seen, they recognize the presence of such objects. But then they are passed on to other activation and pooling layers and on to the next layer of neurons (filters), without recognizing what are the relations *between these objects we identified in that single layer.*
They just account for their presence.

So a *(simplistic)* Neural network will not hesitate about categorizing both these dogs, Pablo and Picasso, as similarly good representations of “corgi-pit-bull-terrier mix”.

Normal (Convolutional) Neural Network will recognize both of these lovely dogs as the same type of dog face, because it does not care about WHERE the elements of the dog's face are located relatively to each other in space. Picasso (the dog on the left) will luckily not be discriminated against by the model, but we really want a model to realize this is not a regular example of a corgi-pit-bull-terrier mix dog.
Image source: The Sun. Image: Lovable Dog Rescue

The network will categorize both these dogs as similarly good representations of "corgi-pit-bull-terrier mix" dogs because they both answer the same conditions at the face composition convolutional layer, for example:

```
if: (2 eyes & pitbullmix_snout
     + pitbullmix_wet_nose & mouth)
then: pitbullmix_face
```

Incorrectly activating the neuron for pitbullmix_face, Instead of something like:

```
if: 2 eyes
& BELOW: pitbullmix_snout
     & pitbullmix_wet_nose
```

```
  & BELOW: mouth
  then: pitbullmix_face
```

Conversely, capsule networks represent directionality along with content and connect between capsules of neurons to infer spatial relationships and retain pose information.

Lacking a grouped capsule representation, pose calculations, and overlapping check between the capsules leads to the next problem.

Problem #3: CNNs can't transfer their understanding of geometric relationships to new viewpoints.

This makes them more sensitive to the original image itself in order to classify images as the same category.

CNNs are great for solving problems with data similar to what they have been trained on. It can classify images or objects within them which are very close to things it has seen before.

But if the object is slightly rotated, photographed from a slightly different angle, especially in 3D, is tilted or in another orientation than what the CNN has seen - the network won't recognize it well.

One solution is to **artificially create tilted representation** of the image or groups and add them to the "training" set. However, this still lacks a fundamentally more robust structure.

Encoding a viewpoint-invariant spatial relationship Pose

So how can we encode spatial relationships between 3D objects?

Hinton took inspiration from a field that already solved that problem: 3D computer graphics.

在三维图形中，位姿矩阵是表示物体间关系的一种特殊技术。体位本质上是表示平移和旋转的矩阵。现在我们知道了。利用子对象间的位姿关系来保持空间关系信息；以4D位姿矩阵的形式测量物体之间的相对旋转和平移。

In 3D graphics, a **pose matrix** is a special technique to represent the relationships between objects. Poses are essentially matrices representing *translation* plus *rotation*. Now we got it. We can retain spatial relationships information using pose relationships between sub-objects; measuring the relative rotations and translations between objects as a 4D pose matrix.

This would be key to understanding dynamic routing between capsules.

. . .

Now that we know the basics of Neural Networks and their spatial recognition problems, we can continue to learn about the recently developed solution to that: Capsule Neural Network. That will be the topic of my next post. *Stay tuned!*