There are two fundamental sub-networks within Convolutional Neural Networks (CNN).

- convolution — local detection of features/patterns
- aggregation/abstraction

A CNN alternates between the two structures.

Diagramming neurons can get complex and does not scale well; we use matrices instead.

We define the parameters of a subnetwork of a neural network as follows:

1. f — the filter size
2. n_c — the number of input channels (matrices)
3. s — the step size; how far to move the filter between iterations
4. p — the padding; extra area around the input edge added to emphasize edge features

Convolution

Consider the example of analyzing a greyscale image; it may look like:

$![convolutionw/f = 3, nc = 36, s = 1, p = 0](https://paper-attachments.dropbox.com/s_A78EEC73035B10A6424111F8CC18$

We pass the filter over each fxf submatrix of the input, performing a column dot product.
We then apply an ReLU function before entering into the output matrix.
The values in the filter are chosen via gradient descent.
We can notice that each of the hyperparameters affect the size of the output.
We choose these parameters heuristically, as many good models already exist.
The output is 4x4x1 in this example, but we could do the following transformations.

- f++ → 3x3x1
- n_c++ → 4x4x2
- s++ → 2x2x1 (in this case some of the input data is ignored)
- p++ → 6x6x1

We can model a convolution with increased parameters like so:

$![complexconvolutionw/f = 3, nc = 3, s = 1, p = 0](https://paper-attachments.dropbox.com/s_A78EEC73035B10A64241111$

This behavior may bee seen in a color (rgb) image seeking to detect two features

Max-Pooling

An example may take the following form:

$![max-poolw/f = 2, nc = 1, s = 2, p = 0](https://paper-attachments.dropbox.com/s_A78EEC73035B10A6424111F8CC18$

The output is determined by the maximum of the values in the region it represents.
This effectively functions to aggregate data values to the most likely representation.
Padding is almost always zero, and there are no weights.

These two operations are applied alternately to form a CNN:

$![](https://paper-attachments.dropbox.com/s_A78EEC73035B10A6424111F8CC18FF4AA14B9B2A8F73FF013C6765837$

As we move right, we tend to increase filter count and thus n_c, but decrease f.
We can thus think of this operation as stretching and squishing volume in alternation.
The number of weights tends to be controlled and independent of the data.
We evaluate these with **tensors** ≡ parallelize-able multidimensional arrays.

The Future of Neural Networks

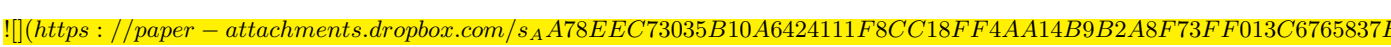
Neural networks are intriguing because they allow us to do new things in a pretty simple way.
BUT people encounter issues:

1. data hunger
   this is self explanatory — neural networks require lots of labeled data to train

2. brittleness/lack of robustness
   small, seemingly arbitrary changes to data can cause large output changes
   this leads to adversarial attacks which seek to exploit brittleness
3. difficulty of explanation
   neural networks are unable to effectively explain why a certain output is correct
   this stems from the fact that NN are model-free and focus on function fitting

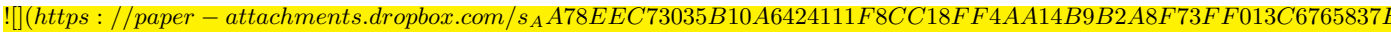
Here is an example of an integrated model developed in Dr. Darwiche's lab: The data below compares the correctness of three approaches to image recognition

1. Bayesian Network
2. Bayesian Network + Background Knowledge
3. Neural Network



We observe that approach (b) beats the neural network handedly in low data situations.
Even in high data situations, approach (b) tends to beat out the neural network.

The below data compares the approaches when trained with different data than tested with.
Noise is either horizontal or vertical pairs, and the models see only one type in each.



In this situation, approach (b) is relatively unaffected, but the neural network falls apart! This is analogous to a student solving problems by applying formulae.
If a query represents something not well represented in the training data, an NN fails.
Though neural networks do very well with prototypical situations, this is impractical.

These type of brittleness errors are common in applications like language translators.

This is okay in those situations, but for higher risk thinks like self-driving cars, it is not.
Thus we can see that AI demands the integration of model-based and -free approaches.