Useful Layers for Deep Learning

Kris Sankaran

Nepal Winter School in AI

December 25, 2018

Outline

Fully Connected Layers

- From previous lecture, think of this as mixture of of logistic regressions

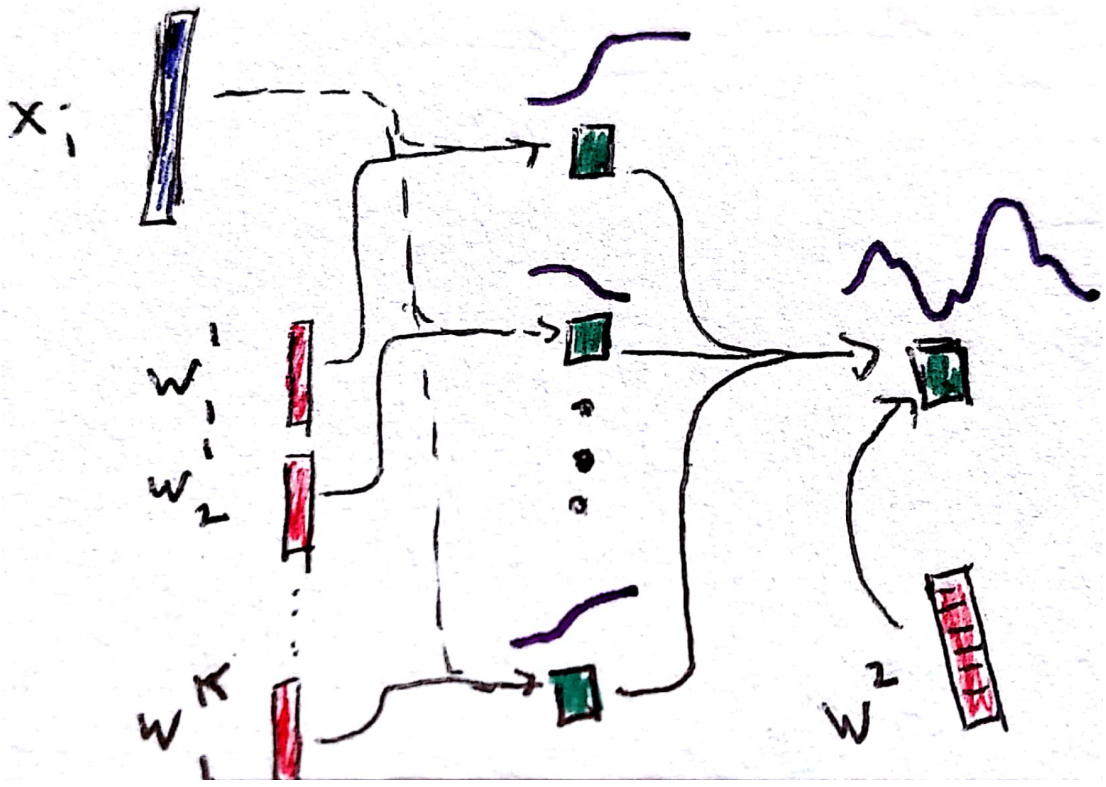


Figure: Mixing more sigmoids, we can represent even more complicated functions.

Nomenclature

- Reason for the name: Every output layer coordinate can see every input layer coordinate
- Biologically motivated shorthand: "layer coordinate" $\rightarrow$ neuron
- The two layers are *fully connected*

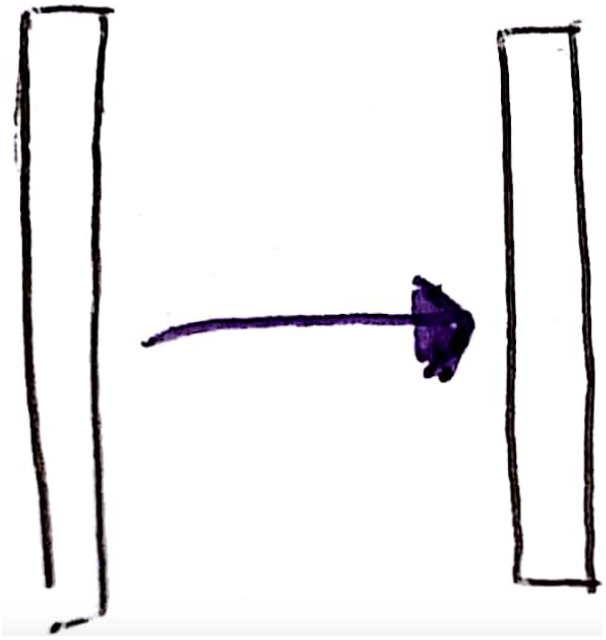


Figure: Our earlier notation for a fully connected layer.

Nomenclature

- Reason for the name: Every output layer coordinate can see every input layer coordinate
- Biologically motivated shorthand: "layer coordinate" $\rightarrow$ neuron
- The two layers are *fully connected*

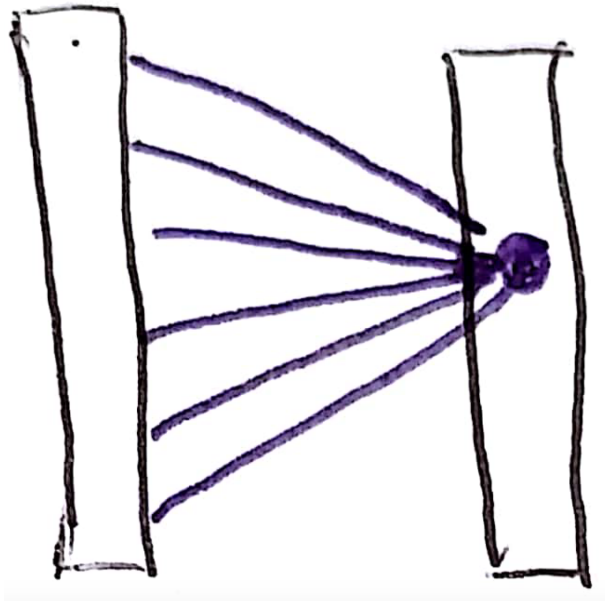


Figure: Each output unit is connected to all of the previous layer's inputs.

Nomenclature

- Reason for the name: Every output layer coordinate can see every input layer coordinate
- Biologically motivated shorthand: "layer coordinate" $\rightarrow$ neuron
- The two layers are *fully connected*

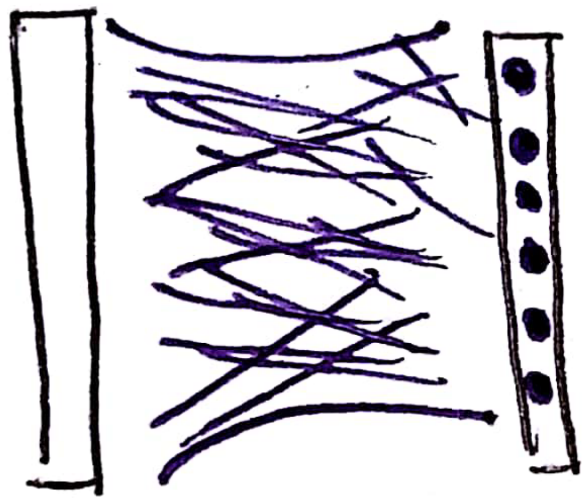


Figure: All the outputs from one layer can affect all the inputs to the next one.

Fully Connected Layers: Formula

- This has concise mathematical notation

$$\sigma\left(W_k h_{k-1}\right)$$

- The nonlinearity σ is applied elementwise
- It's common to explicitly include a bias term b_k inside the nonlinearity: $\sigma\left(W_k h_{k-1} + b_k\right)$

Outline

The Probability of Sums

- First a digression on 1D sequences
- Material largely taken from Chris Olah's (incredible!) blog (`http://colah.github.io/`)

The Probability of Sums

- Drop a ball and see where it lands
  - Drop it slightly to the right (for reasons that will be clear later)
- For simplicity, suppose it lands on some grid
- Consider the probabilities of different outcomes

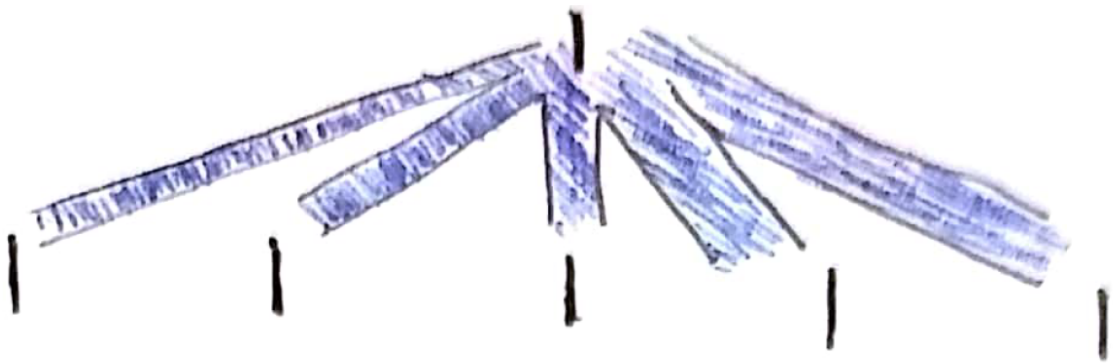


Figure: Branch widths correspond to the probabilities of landing in different positions.

The Probability of Sums

- Drop it again, starting from the place it fell last time
- Each branch gives a path from original to final position
- Branch probability $\rightarrow$ product of original probabilities
- E.g., go right by 2 steps then left by 1 is $f(2) f(-1)$

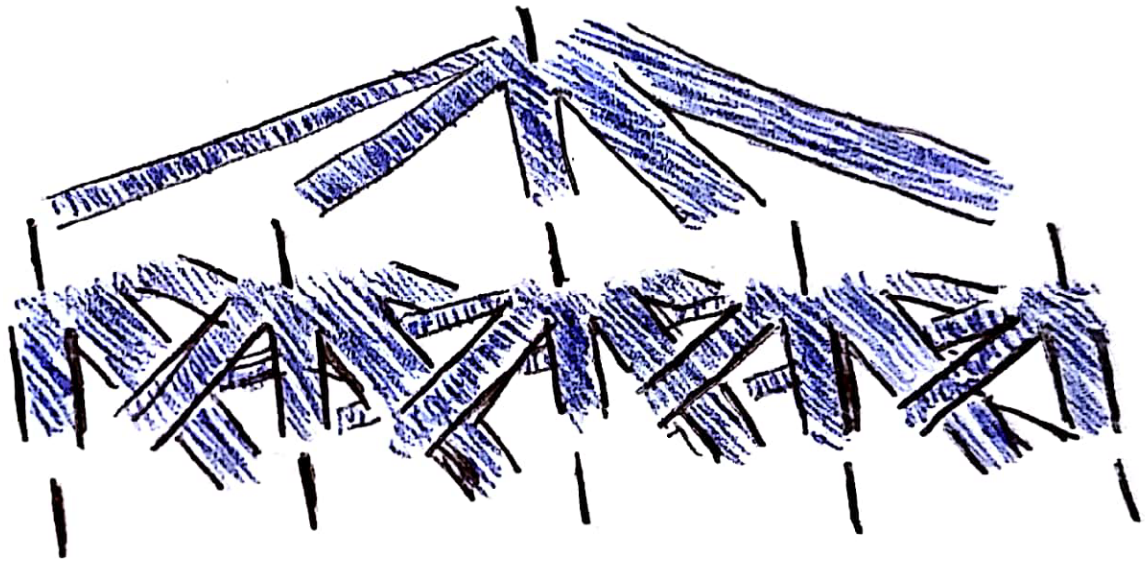
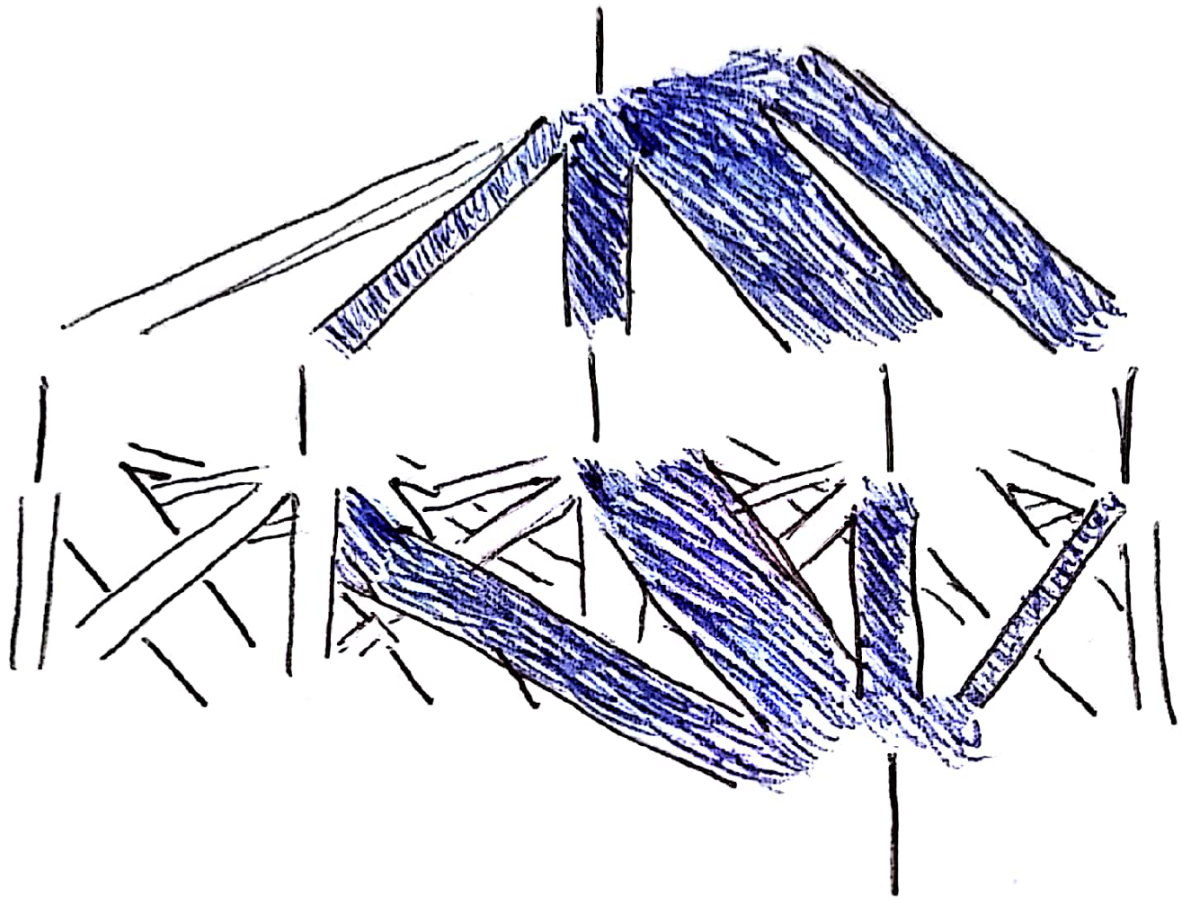


Figure: Can compute probabilities of landing in different positions, after two drops.

The Probability of Sums

- Probability of landing at a prespecified position after two drops?
- Consider any of the intermediate positions it could have landed at
- Sum over all the paths that lead to the same destination



The Probability of Sums

- Probability of landing at a prespecified position after two drops?
- Consider any of the intermediate positions it could have landed at
- Sum over all the paths that lead to the same destination

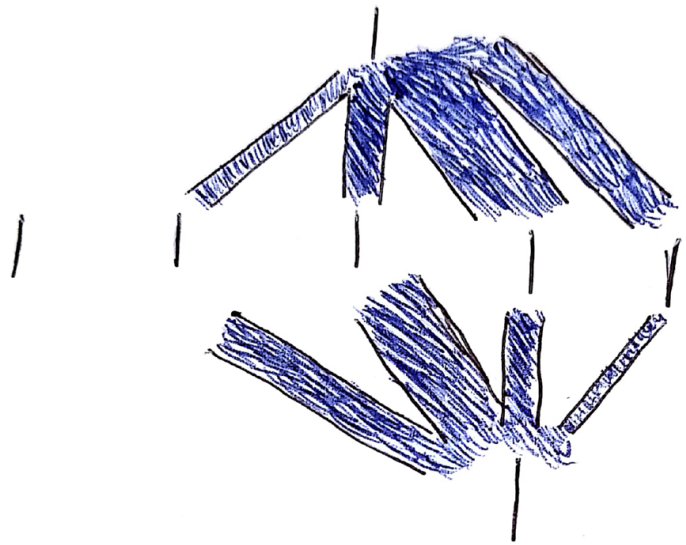


Figure: Each branch starts at the initial dropping position and ends at u.

Mathematical Formula

- This can be expressed concisely

$$\mathbb{P}(X+Y=u) = \sum_{k=-\infty}^{\infty} f(k) f(u-k)$$
$$\overset{\text{defined}}{=} f * f(u)$$

- Notice that the upside-down tree is reflected and translated
  - Exactly the transformation $f(x) \to f(u-x)$

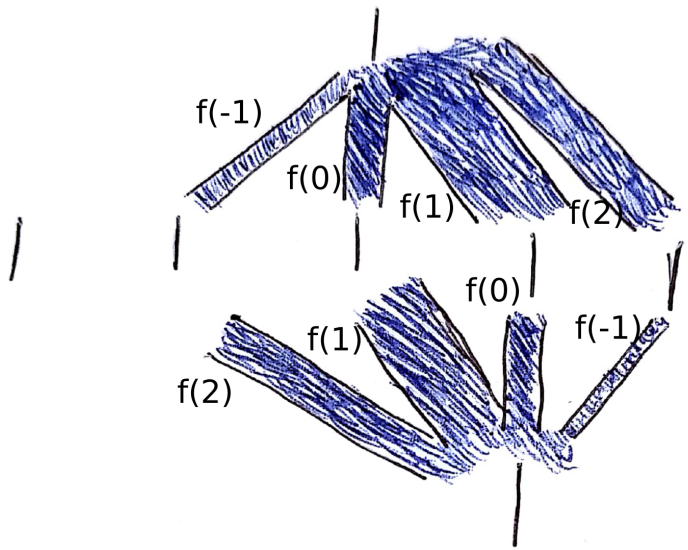


Figure: Each branch starts at the initial dropping position and ends at u.

Varying functions

- We didn't need to drop in the same way the second time

$$\mathbb{P}(X+Y=u) = \sum_{k=-\infty}^{\infty} f(k)\, g(u-k)$$

$$\overset{\text{defined}}{=} f * g(u)$$

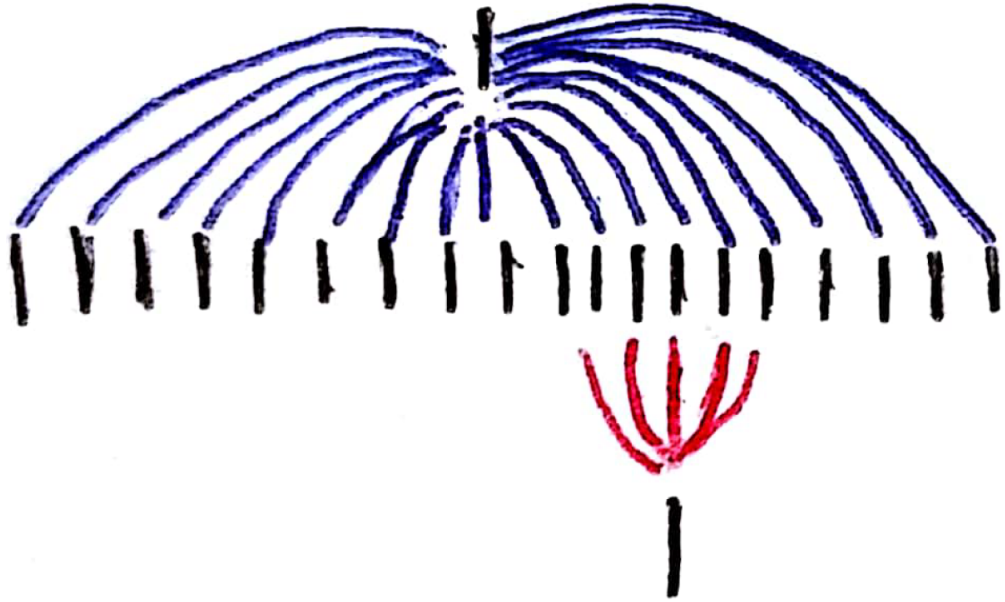


Figure: Suppose we first dropped from very high, and then afterwards from very low, so that it can't move very far from its second position.

Outline

Convolution and Computer Vision

- Convolutions have been very successful in computer vision
- Many reasons,
  - Weight sharing / local connectivity means far fewer parameters
  - Can be computed very fast, especially on GPUs
  - Related to biological vision

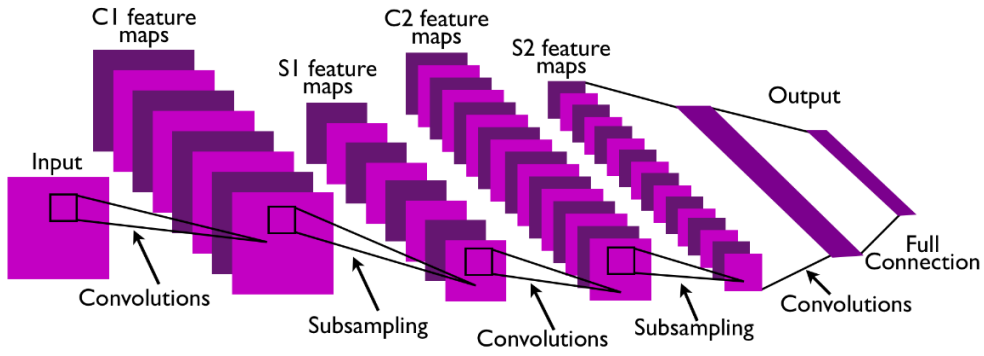


Figure: A typical computer vision architecture. Note the convolution layers.

Image Convolution

- Our previous discussion applies for dropping a ball onto a 2D surface
- Think of an image as a function over a grid
- The function for the second drop designed to have small support
  - Typically called "filters"
  - These will be learned from data

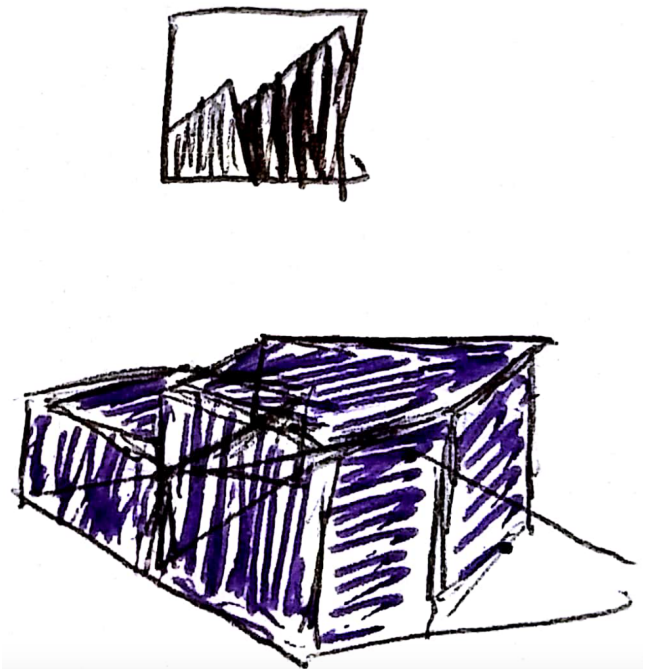


Figure: Think of an image as a function in 2D. Each pixel would be the endpoint for a branch in the "dropping ball" example we've been working with.

Example Computation

Check interactive example: `http://setosa.io/ev/image-kernels/`

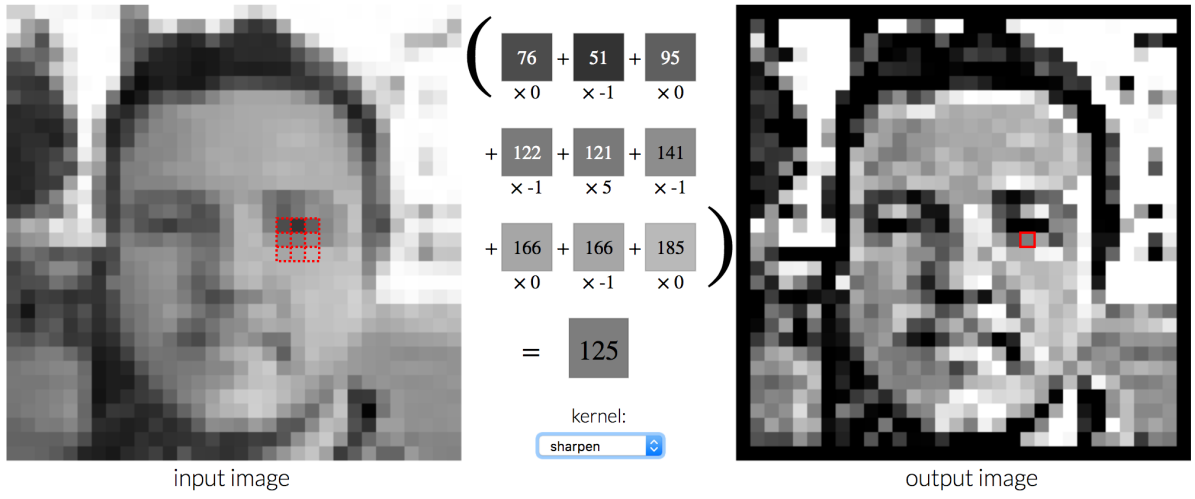


Figure: An example convolution with a single (sharpening) filter.

Blurring

- You can learn many types of image processing operations using prespecified filters
- Neural networks *learn* filters that are ideal for given classification tasks

0	0	0	0	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	0	0	0	0

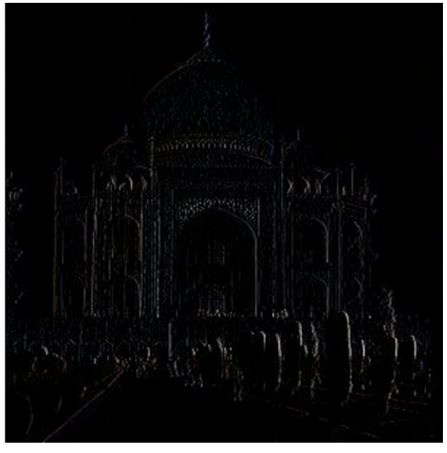
Derived from the Gimp documentation

Figure: Blurring can be implemented by convolving with a small plateau.

Edge Detection

- You can learn many types of image processing operations using prespecified filters
- Neural networks *learn* filters that are ideal for given classification tasks

0	0	0	0	0
0	0	0	0	0
0	-1	1	0	0
0	0	0	0	0
0	0	0	0	0



Derived from the Gimp documentation

Figure: Edges can be found by convolving with a small zigzag.

Local Connectivity

- Convolutions have small support
- Connections between layers are sparse
- More biologically plausible than having all neurons connected with one another

Figure: After stacking the image into a long vector, A convolution layer creates a local connectivity pattern between layers (contrast with fully connected layers).

Max Pooling

- Doing some sort of downsampling can help in various applications
- Most common: Take only largest value from a small patch
  - Other options are possible, e.g., averaging

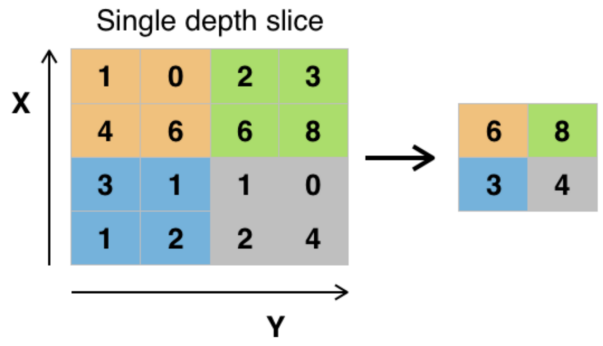


Figure: Max pooling reduces the size of a feature map by taking only the largest value within patch.

Some variants

- Different types of convolution are useful in different subfields of deep learning
  - Causal convolution [8]
  - Dynamic Convolutions [4]
  - Atrous convolution [2]
  - Deconvolution [6]
- Not just computer vision!

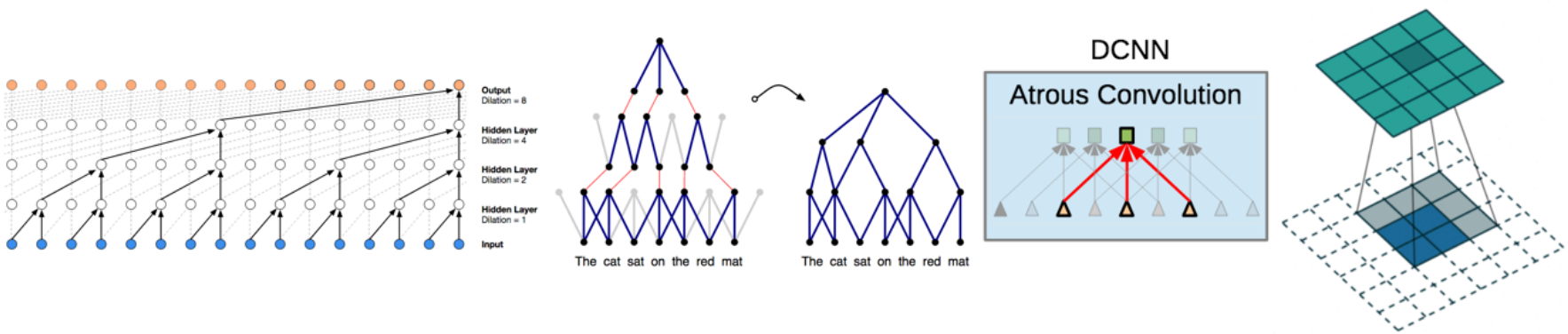


Figure: Examples of the four types of convolutions above.

Outline

Residual Blocks

- Deeper layers learn more meaningful representations
- Empirically, can be much harder to train
- What to do?

Learning Residuals

- Introduce complexity on top of existing approximation
- Try to transform the previous layer in an intelligent way
- Has a feel of forwards stagewise fitting or boosting, and is inspiration for deep ODEs

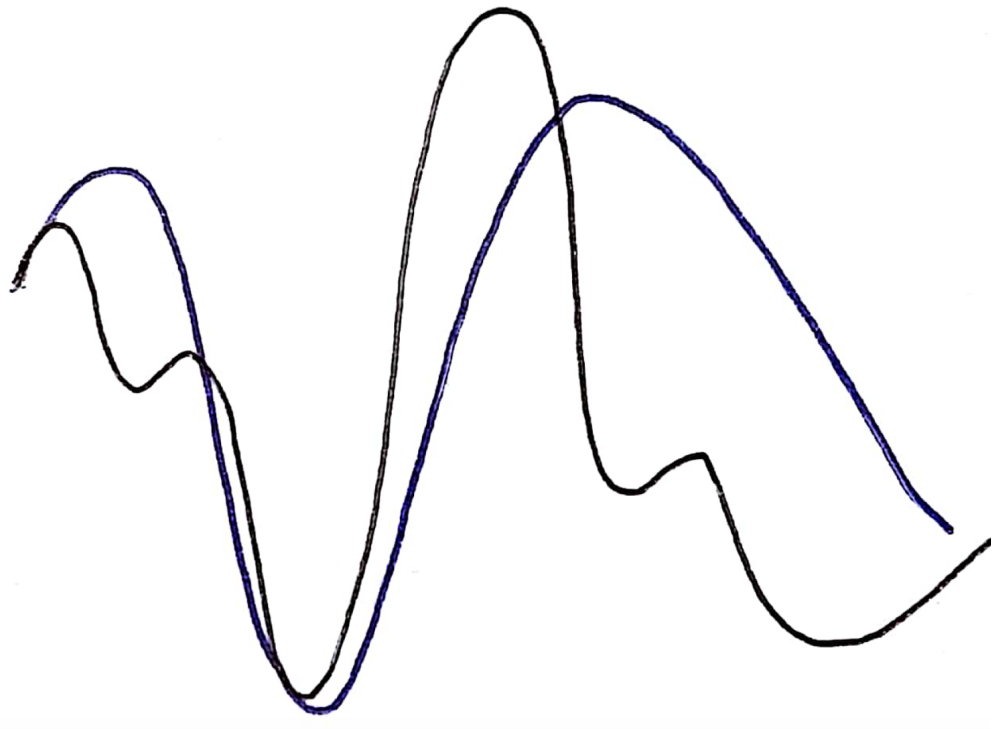


Figure: An example of the difference between the true and the observed

Learning Residuals

- Introduce complexity on top of existing approximation
- Try to transform the previous layer in an intelligent way
- Has a feel of forwards stagewise fitting or boosting, and is inspiration for deep ODEs

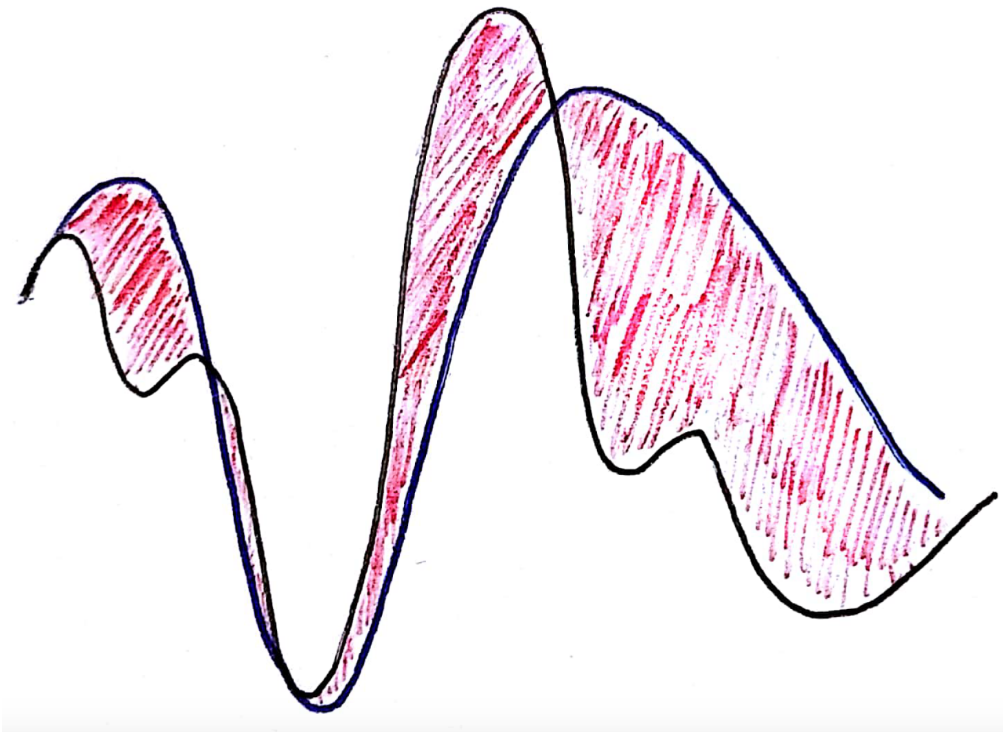


Figure: An example of the difference between the true and the observed

Learning Residuals

- Introduce complexity on top of existing approximation
- Try to transform the previous layer in an intelligent way
- Has a feel of forwards stagewise fitting or boosting, and is inspiration for deep ODEs

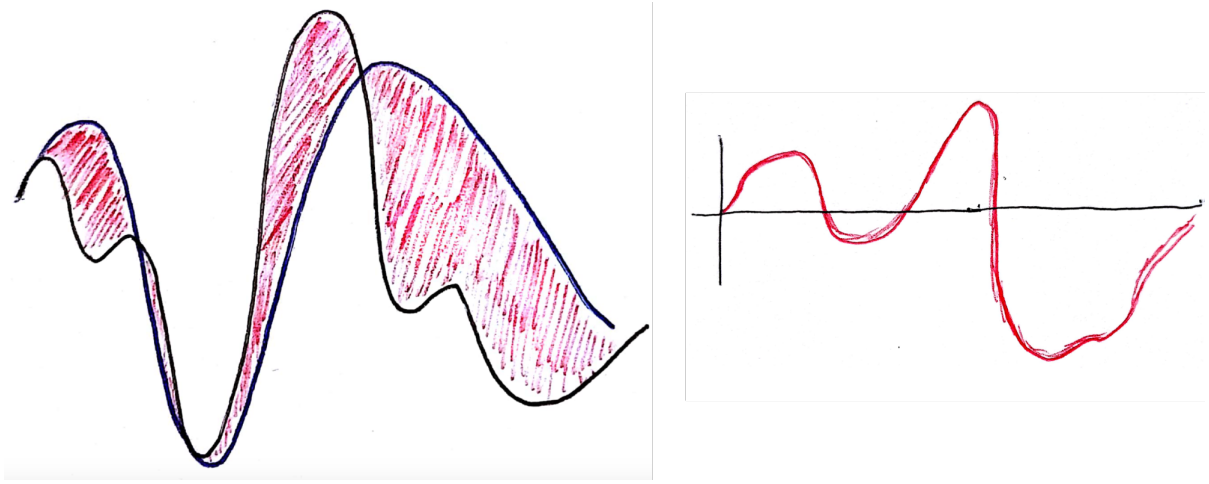


Figure: We will attempt to directly learn the residual (in red), rather than trying to arbitrarily modify the current approximation.

A First Attempt

- Introduce a direct connection $h_{k-1} \to h_k$

$$h_k = f^k\left(h_{k-1}, W_k\right) + h_{k-1}$$

- Issues?

A First Attempt

- Introduce a direct connection $h_{k-1} \to h_k$

$$h_k = f^k\left(h_{k-1}, W_k\right) + h_{k-1}$$

- Issues?
  - Can't adapt different mixing weights on residual
  - Input and output dimensions might not agree

Improved Architecture

- No longer a direct addition, but encourages focusing on residual

$$h_k = W^r_{k,1} f^k \left(h_{k-1}, W_k\right) + W^r_{k,2} h_{k-1}$$

- If input and output dimensions match, don't need $W^r_{k,2}$

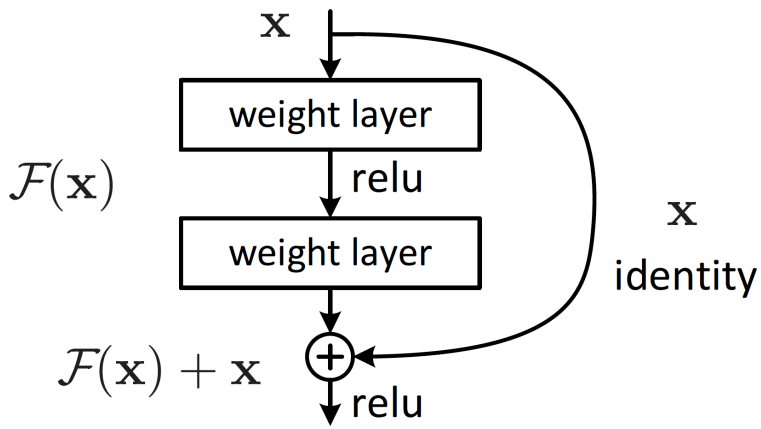


Figure: Resnet building block, according to the improved architecture.

Applications

- Facilitates training for very deep networks
- Used successfully in several competitions

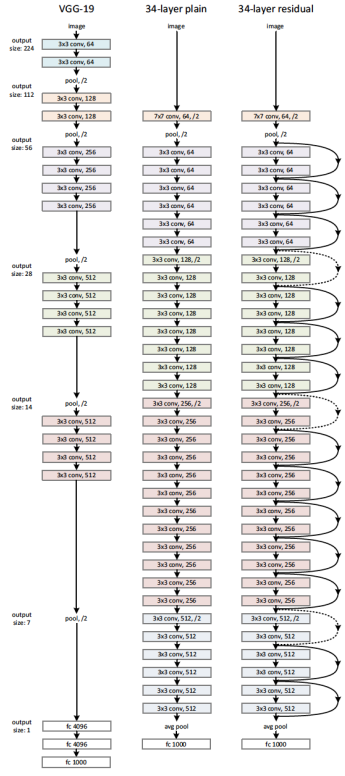


Figure: The middle column escribes a naive approach to a very deep network, which fails to train. The residual network (right) is better than the other two.

Conclusion

- Modularity in backprop $\rightarrow$ design independent layers
- Convolution and residual layers enjoy empirical success in range of domains
- Many types of layers that we haven't discussed
  - Sequential data: LSTM Cells [7], Gatted Recurrent Units [3]
  - Attention mechanisms [1]
  - Feature-wise Modulation [5]

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[4] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[5] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[7] Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.