Neural Networks continued

Martha Lewis

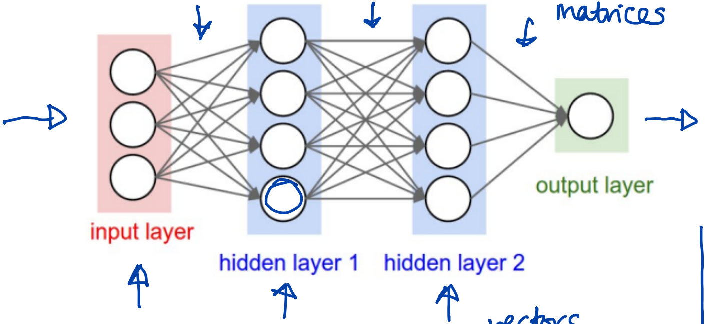
Department of Engineering Mathematics
University of Bristol

Outline

- Convolutional Neural Networks
- Recurrent Neural Networks - GRUs, LSTMs
- Autoencoders
- Generative Adversarial Networks

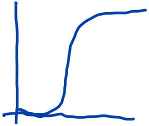
Feed-Forward Neural Networks

matrices



label

vectors



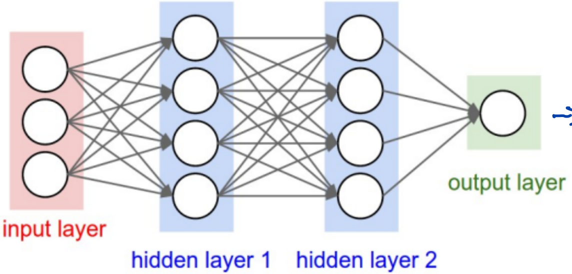
- Forward pass:

$$h_j = \sigma\left(\sum_i W_{ji} x_i\right) = \sigma(W\vec{x})$$

where σ is some kind of non-linearity (logistic sigmoid, ReLU, tanh, ...)

Image credit:
`https://brilliant.org/wiki/feedforward-neural-networks/`

Feed-Forward Neural Networks

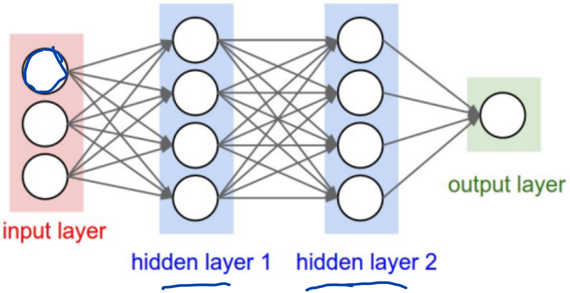


→ $\underbrace{\text{pred} \quad \text{label}}_{\text{loss function}}$

- Backward pass: Update weights by backpropagation of error through the network

Image credit:
`https://brilliant.org/wiki/feedforward-neural-networks/`

Feed-Forward Neural Networks



- When layers are fully connected these are sometimes called dense layers (Tensorflow, Keras)
- Deep Neural Networks have many hidden layers, but they are not always dense layers and often of different kinds

Image credit:
`https://brilliant.org/wiki/feedforward-neural-networks/`

Convolutional Neural Networks

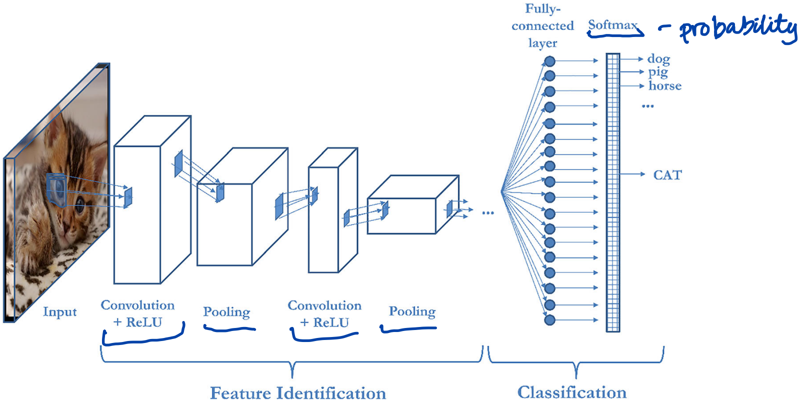


FIGURE 2 The generic processing flow of a deep convolutional neural network, whose architecture involves transforming input signals through many sequences of locally-connected convolutional, ReLU, and pooling nodes, before finally passing them to a fully connected classification layer that assigns final category labels

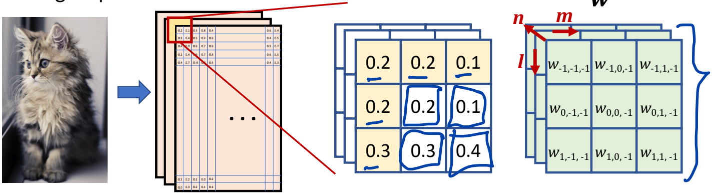
Image credit: Buckner, Cameron. “Deep learning: A philosophical introduction.” Philosophy Compass 14.10 (2019)

Convolutional Neural Networks

Convolutional layer

$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$



This is repeated for every pixel of the image (if *stride*=1)

Image is padded for the bordering pixels

So that the output is the same size as its input

Image credit: Farnoosh Heidarivincheh

Convolutional Neural Networks

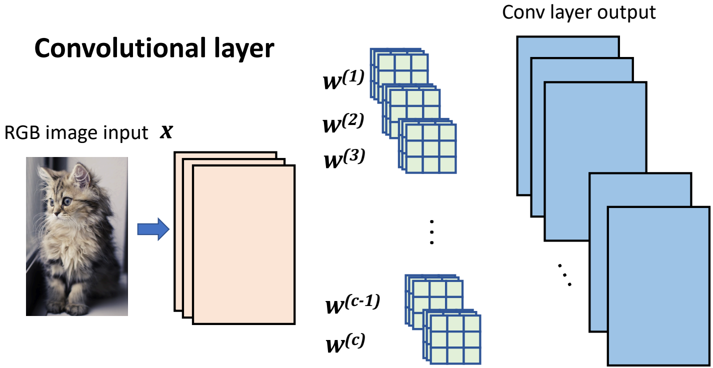
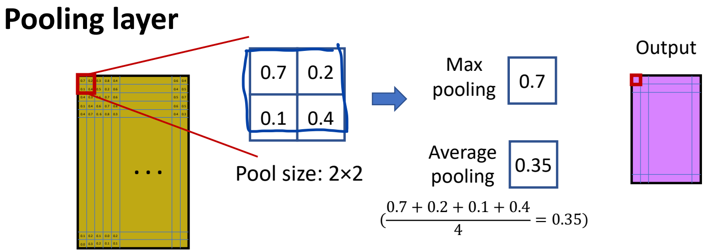


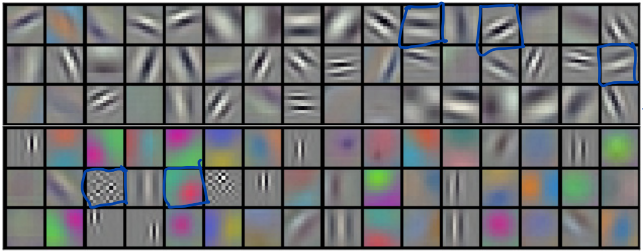
Image credit: Farnoosh Heidarivincheh



$$\left(\frac{0.7 + 0.2 + 0.1 + 0.4}{4} = 0.35\right)$$

Image credit: Farnoosh Heidarivincheh

Convolutional Neural Networks



1-48

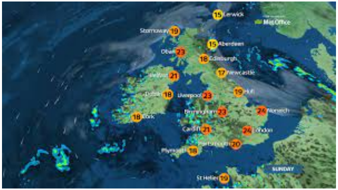
49-96

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." Communications of the ACM 60.6 (2017): 84-90.

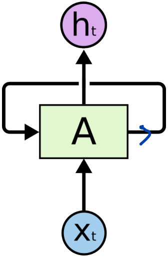
ConvNet tutorial

Try working through `https://www.tensorflow.org/tutorials/images/cnn` - this will also be part of this week's worksheet

Recurrent Neural Networks



Recurrent Neural Networks



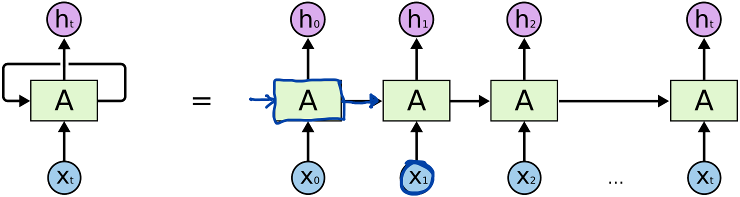
- Instead of just feeding forward, the output of the network is fed back in the next timestep

$$h_t = \sigma(A \cdot [h_{t-1}, x_t] + b)$$

where $[h, x]$ indicates the concatenation of these vectors

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

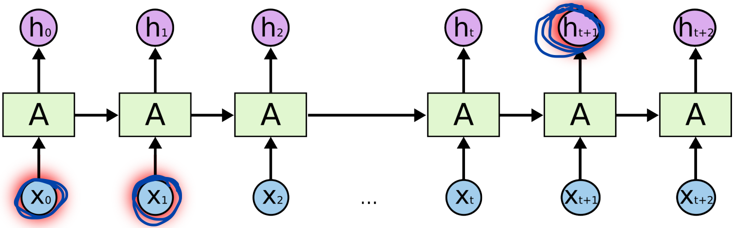
Recurrent Neural Networks



- Training is via *backpropagation through time* (BPTT)
- Standard backpropagation, keeping weights equal at each layer

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

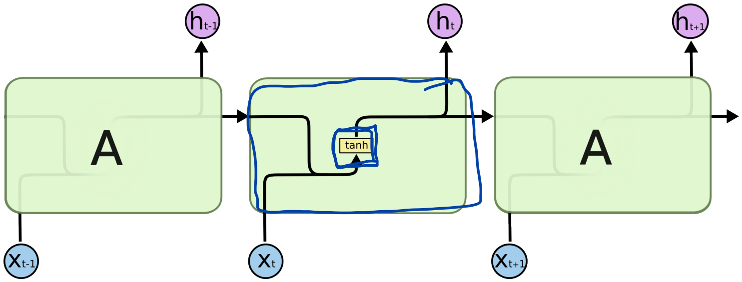


- When errors backpropagate through a large number of layers, the gradients can become very small

Bengio et al 1994 vanishing gradients.

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

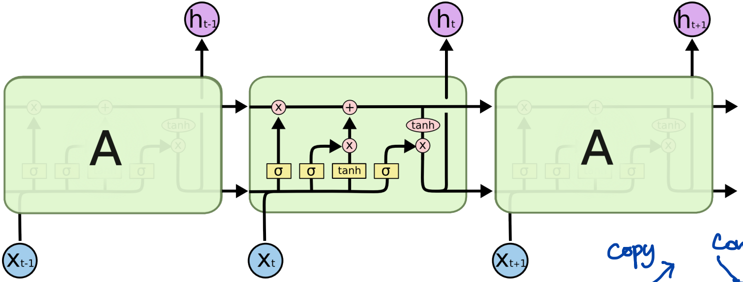
Recurrent Neural Networks



- The interior of each cell looks the same.

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory Networks



Copy

Concatenation

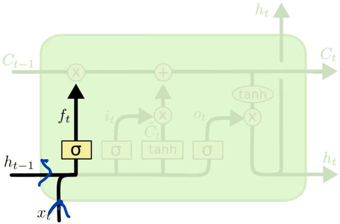
- LSTMs have a more complex interior architecture which allows certain states to be remembered for longer.

Hochreiter & Schmidhuber 1997

---

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

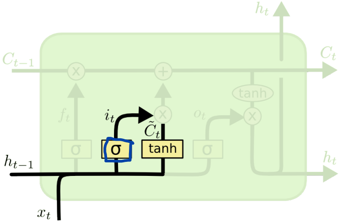
Long Short-term Memory Networks



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

- The first part is the 'forget' gate, which decides on part of the input to forget

Long Short-term Memory Networks



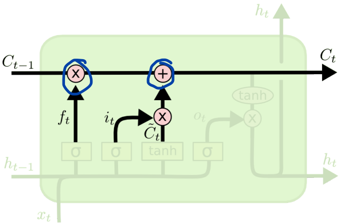
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- The input gate decides which values to update, and new candidate cell values are created

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory Networks

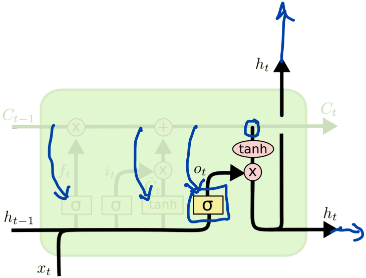


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- The cell state is updated

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory Networks



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- Finally the output gate decides which values to output.

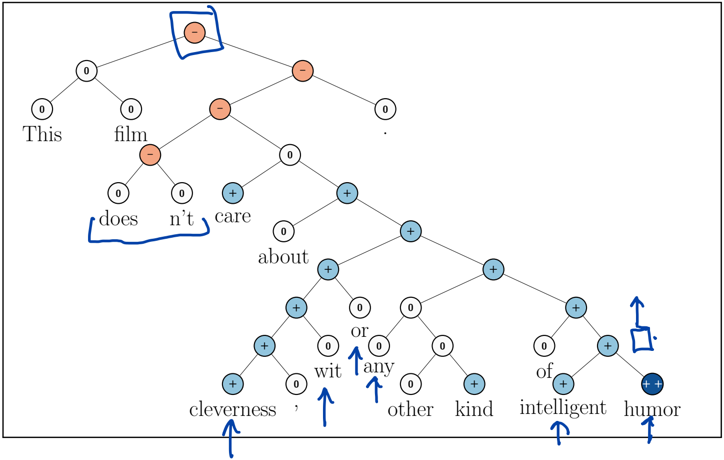
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory Networks

- Alternatives to LSTMs include introducing ‘peephole connections’ which allow the gates to access the cell state as well as the input
- Gated Recurrent Units (GRUs) have a simpler gate structure that is still very effective

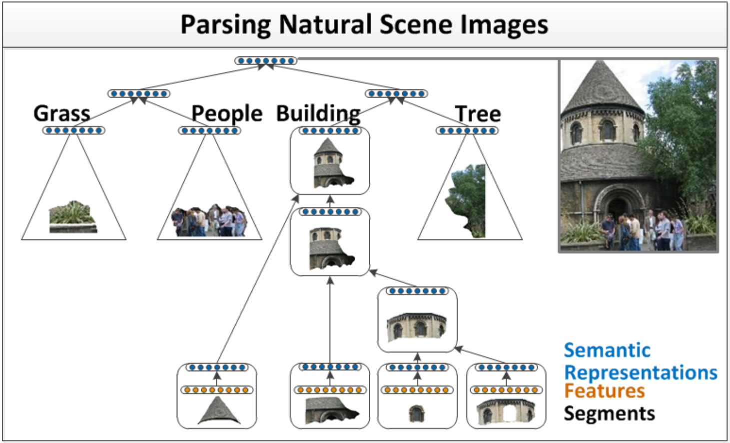
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recursive Neural Networks



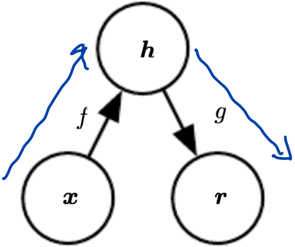
Socher, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP 2013.

Recursive Neural Networks



Socher, Richard, et al. "Parsing natural scenes and natural language with recursive neural networks." ICML 2011.

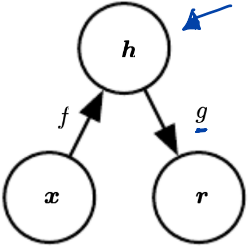
Autoencoders



- Neural network that is trained to copy its input to its output
- Can be considered a special case of a feedforward network
- Two parts:
  - Encoder function $h = f(x)$
  - Decoder function $r = g(h)$

https://www.deeplearningbook.org/contents/autoencoders.html

Autoencoders



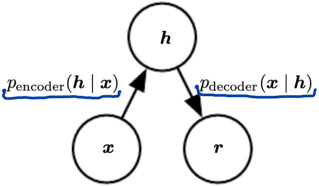
- Learning process: $L(x, g(f(x)))$ = $\|x - g(f(x))\|$
- When the hidden layer h has a lower dimension than x, the autoencoder is said to be *undercomplete*.
- When the decoder is linear and L is MSE, an undercomplete autoencoder learns to span the same subspace as principal component analysis

---

https://www.deeplearningbook.org/contents/autoencoders.html

Types of autoencoder

- Sparse: $L(x, g(f(x))) + \Omega(h)$
- Denoising autoencoders: $L(x, g(f(\tilde{x})))$, where $\tilde{x}$ is a noisy version of x.
- Stochastic autoencoder:



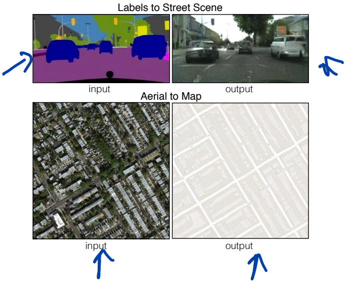
- Examples: `https://www.tensorflow.org/tutorials/generative/autoencoder`

---

https://www.deeplearningbook.org/contents/autoencoders.html

Generative Adversarial Networks

- Why generative modelling?
  - Creative aspect to AI
  - System can create its own output and also do this in sync with a human



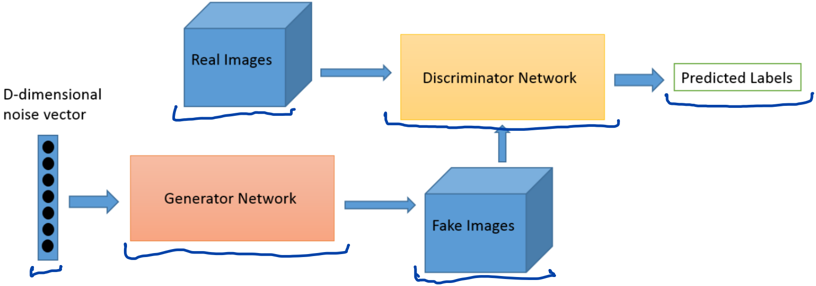
Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016). Image credit: Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004

Generative Adversarial Networks

- Why generative modelling?
  - Creative aspect to AI
  - System can create its own output and also do this in sync with a human
- https://thispersondoesnotexist.com/
- Suwajanakorn, S., Seitz, S. M., & Kemelmacher-Shlizerman, I. (2017). Synthesizing Obama: learning lip sync from audio. ACM Transactions on Graphics (ToG), 36(4), 1-13. https://www.bbc.co.uk/news/av/technology-40598465

Generative vs Discriminative Models

- Given some data, a discriminative model will try to assess the probability of a label given that data: maximize $p(label|data)$
- Given a label, a generative model will try to generate data that fits the label: maximize $p(data|label)$



Training

- The network is trained using two loss functions: $\underbrace{L_G(\theta_D, \theta_G)}$ and $\underbrace{L_D(\theta_D, \theta_G)}$.
- The generative model wants to minimise $\underbrace{L_G(\cancel{\theta_D}, \theta_G)}$
- The discriminative model wants to minimise $\underbrace{L_D(\theta_D, \cancel{\theta_G})}$
- But each only has access to its own parameters

Training

- Analysed game-theoretically - the solution is a Nash equilibrium (θ_D, θ_G) that is a local minimum of L_G wrt θ_G and L_D wrt θ_D
- The generator G and discriminator D are modelled using deep neural networks.
- The neural network can have different structures
- Worked example `https://www.tensorflow.org/tutorials/generative/dcgan`