Introduction to Causal Machine Learning

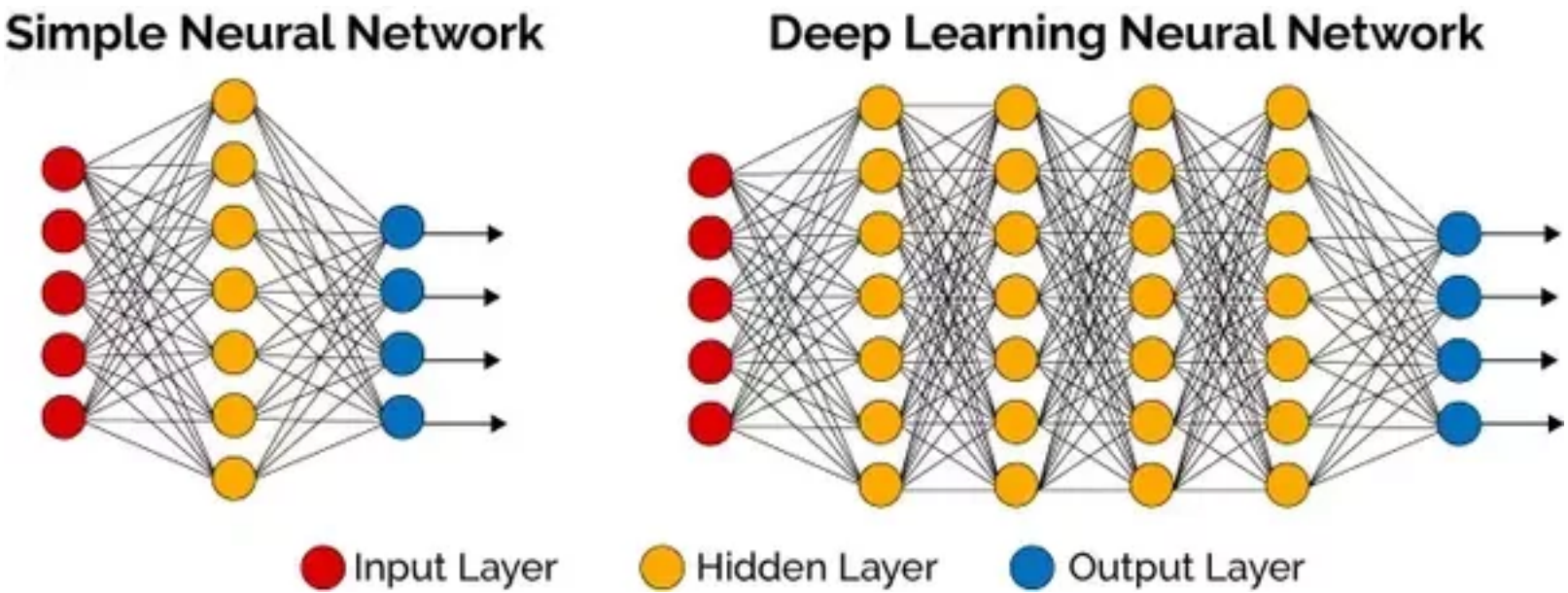
Deep Learning

Anthony Strittmatter

Literature

- James, Witten, Hastie, and Tibshirani (2023): "An Introduction to Statistical Learning", Springer, download,
  Chapter 10.
- Goodfellow, Bengio, and Courville (2016): "Deep Learning", MIT Press, download.
- Gurney (1997): "An Introduction to Neural Networks", UCL Press, download.

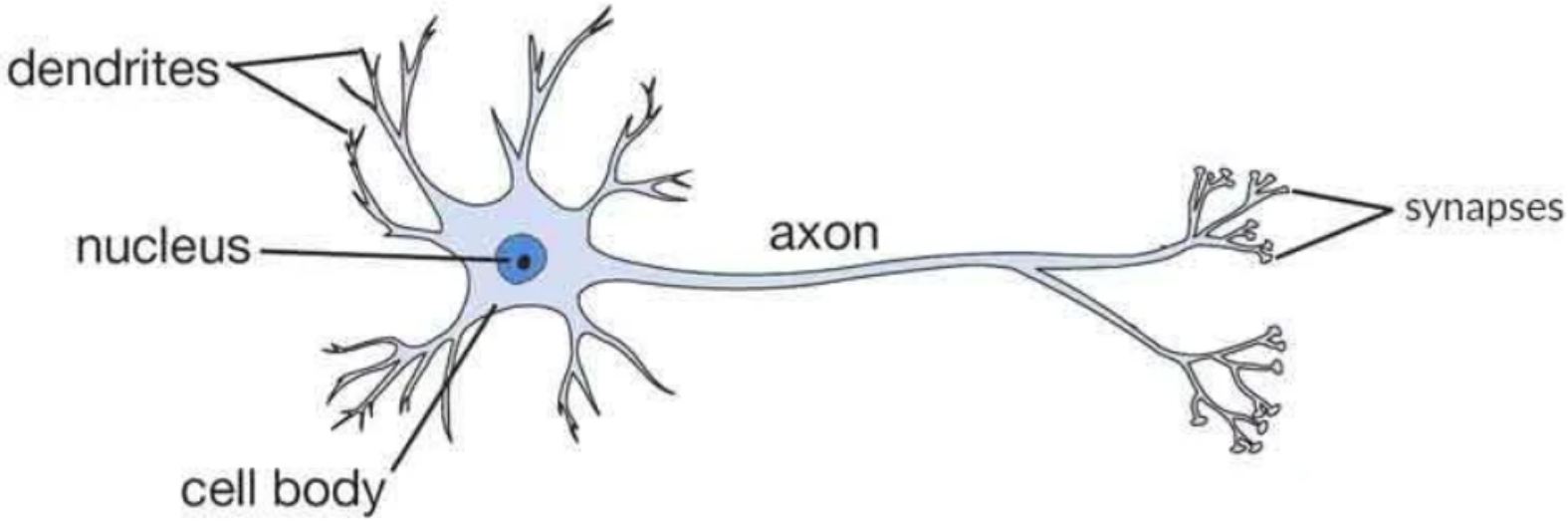
Artificial Neural Networks (ANN)



Basic Idea of Neural Networks

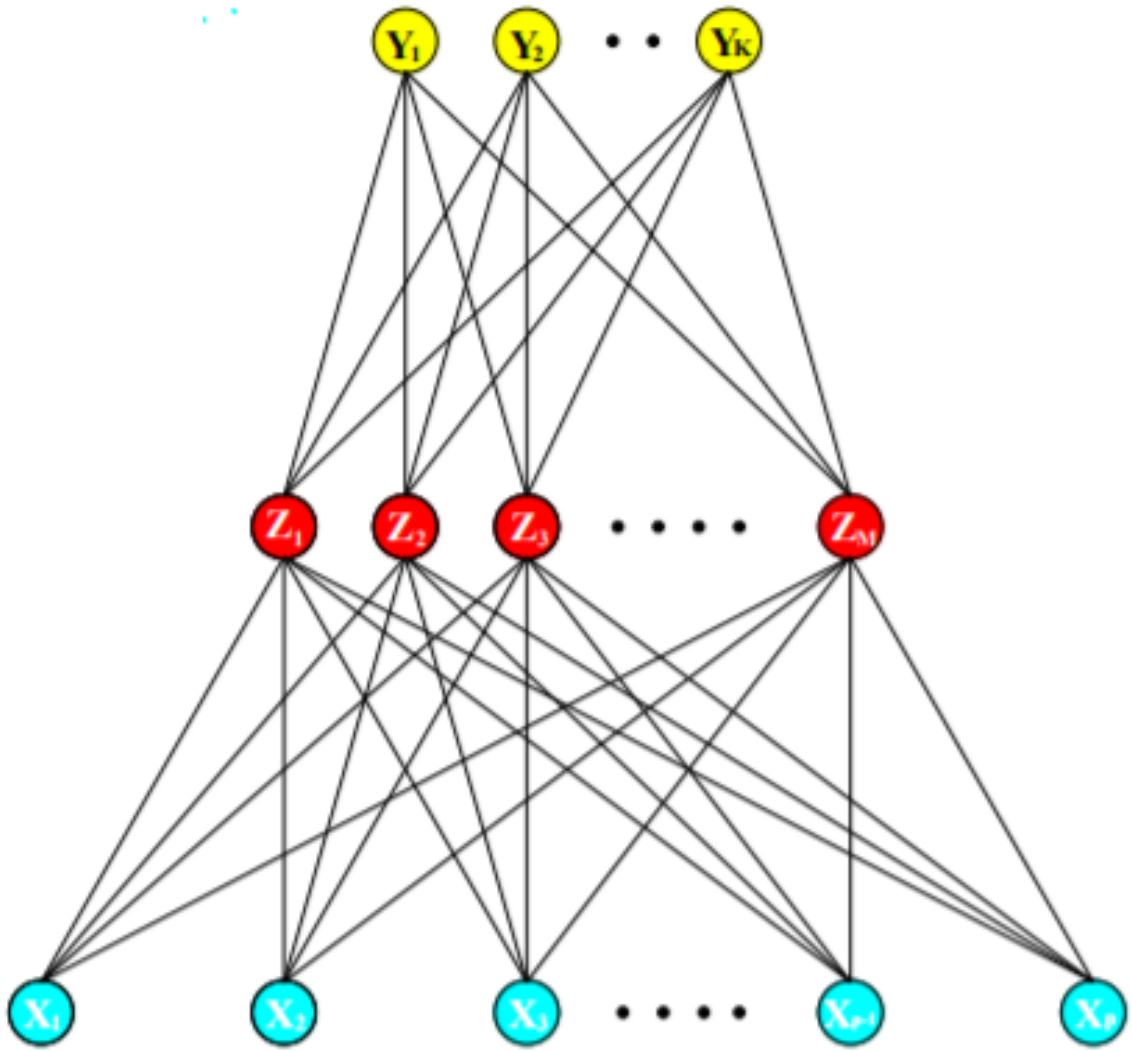
- Neural networks are complex (non-linear) adaptive systems
- Adaptive means it has the ability to change its internal structure by adjusting weights of inputs
- Neural networks were first developed to model human brains
- They are composed of a large number of highly interconnected processing elements known as the neurons
- The connections between neurons represent synapses
- Early neurons fired when the signal passed a certain threshold, which was mimicked by step-functions
- Nowadays, neurons use smoother threshold functions like the sigmoid

Biological Neuron



- Human brains consist of billions of neural cells that process information
- Each neural cell considered a simple processing system
- Dendrites of a neuron receive input signals from another neuron
- Based on those inputs, fire an output signal via an axon

Single Hidden Layer Network



Source: Hastie, Tibshirani, and Friedman (2009)

Two Step Linear Regression or Classification Model

- $X = (X_1, ..., X_P)$ are the **input** layers
- $Z = (Z_1, ..., Z_M)$ are the **neurons or hidden units** of the hidden layer (here only 1 hidden layer)
- $Y = (Y_1, ..., Y_K)$ are the **output** layers
- Predicted output $f_k(X) = (f_1(X), ..., f_K(X))$

$$Z_m = \alpha_{0m} + X\alpha_m \text{ for all } m = 1, ..., M,$$
$$f_k(X) = \beta_{0k} + Z\beta_k \text{ for all } k = 1, ..., K$$

- $\alpha_m = (\alpha_{1m}, ..., \alpha_{Pm})'$ and $\beta_k = (\beta_{1k}, ..., \beta_{Mk})'$ are coefficient vectors (called weights)
- α_{0m} and β_{0k} are intercepts (called bias terms)

Two Step Linear Regression or Classification Model

Merging the two equations:

$$f_k(X) = \beta_{0k} + (\alpha_{01} + X\alpha_1)\beta_{1k} + ... + (\alpha_{0M} + X\alpha_M)\beta_{Mk}$$

$$f_k(X) = \underbrace{\beta_{0k} + \sum_{m=1}^{M} \alpha_{0m}\beta_{1k}}_{=\gamma_0} + \sum_{m=1}^{M} X \underbrace{\alpha_m\beta_{mk}}_{=\gamma_{mk}}$$

$$f_k(X) = \gamma_0 + \sum_{m=1}^{M} X\gamma_{mk}$$

$\Rightarrow$ Breaks down to a simple linear model

Single Hidden Layer Network

Neural networks are non-linear modelling tools:

- $\sigma(\cdot)$ activation function
- $g_k(\cdot)$ is the outcome transformation function

$$Z_m = \sigma(\alpha_{0m} + X\alpha_m) \text{ for all } m = 1, ..., M,$$
$$f_k(X) = g_k(\underbrace{\beta_{0k} + Z\beta_k}_{=T_k}) \text{ for all } k = 1, ..., K$$

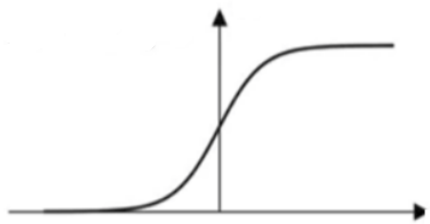
Merging the two equations:

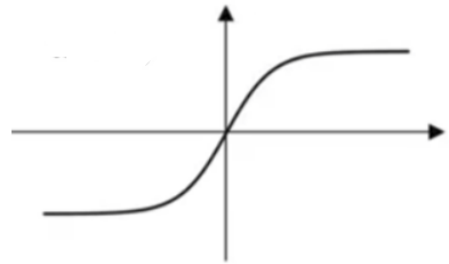
$$f_k(X) = g_k(\beta_{0k} + \sigma(\alpha_{01} + X\alpha_1)\beta_{1k} + ... + \sigma(\alpha_{0M} + X\alpha_M)\beta_{Mk})$$

Popular Activation Functions

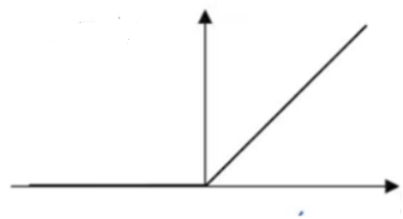
- **S-shaped functions** use often the sigmoid or the hyperbolic tangent transformation:

$$\sigma(v) = \frac{1}{1 + e^{-v}} \quad \text{or} \quad \sigma(v) = \tanh(v) = \frac{\sin(v)}{\cos(v)}.$$





- **Rectified linear unit (ReLU) functions** map negative inputs to 0 and positive inputs to the same output, $\sigma(v) = max(0, v)$.



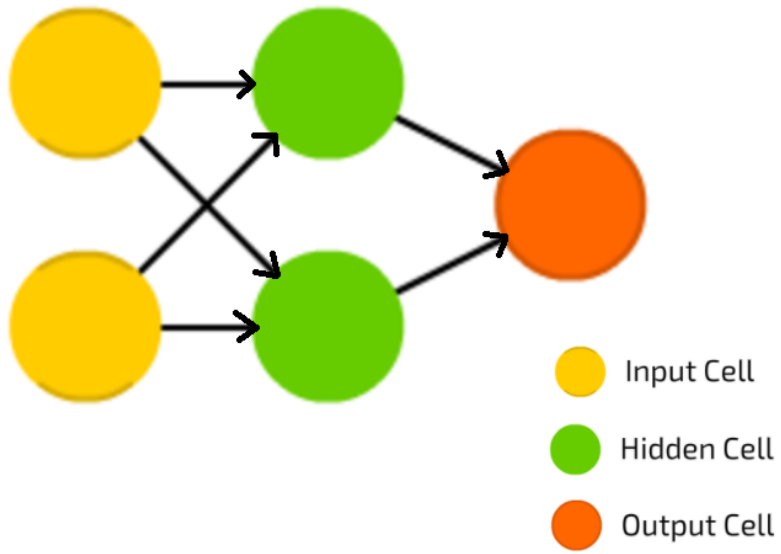
Outcome Transformation Function

- The output transformation allows to account for the distribution of the output (e.g., continuous or binary).
- This is relevant to distinguish between regression and classification problems.
- For regression problems we typically choose the identity function, $g_k(T_k) = T_k$.
- For classification problems the *softmax* function

$$g_k(T_k) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$$

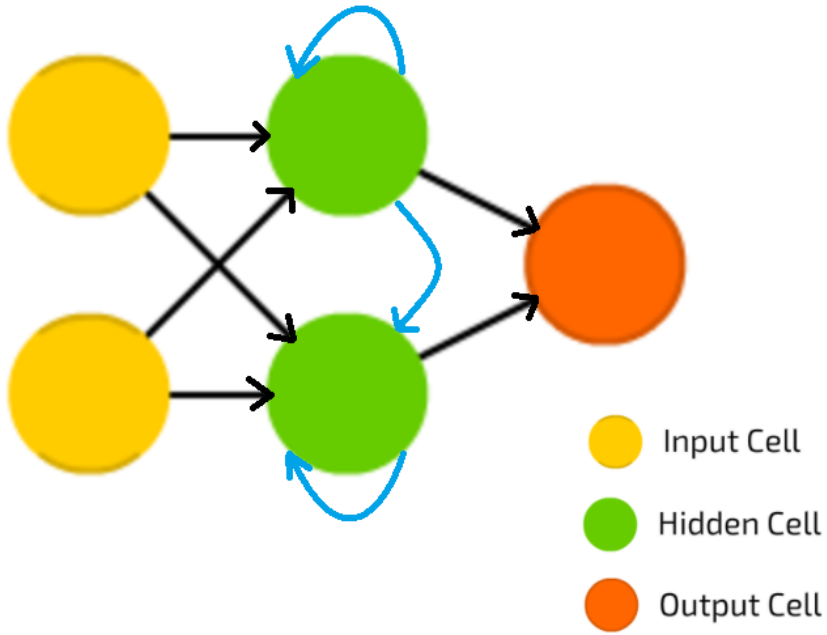
is often used. This is the same transformation that is used for the multinomial Logit. It produces positive estimates that sum to one across the K outcomes (e.g. categorical outcome variables).

Feed Forward Neural Network



- Connections between the nodes do not form a cycle.
- Activation flows from input layer to output, without back loops.

Recurrent Neural Network (RNN)

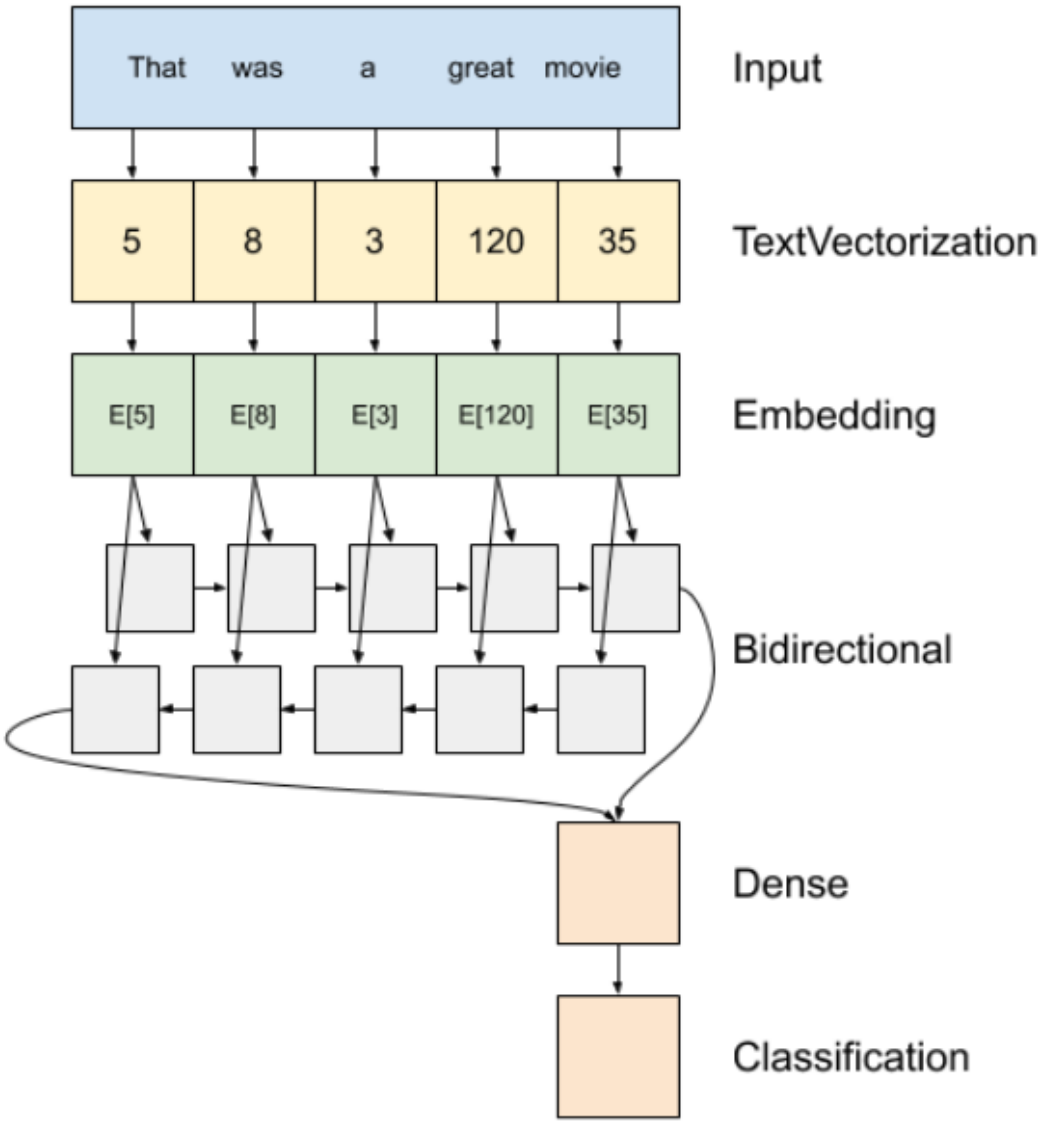


- RNN have connections to between hidden units of the same or previous layers (often with a temporal lag).
- This allows model different aggregation levels jointly or to "remember" the previous input.
- Accordingly, RNNs are suited to model data along a temporal sequence (or sequentially arriving data), e.g. words in a text, time series, voice records.

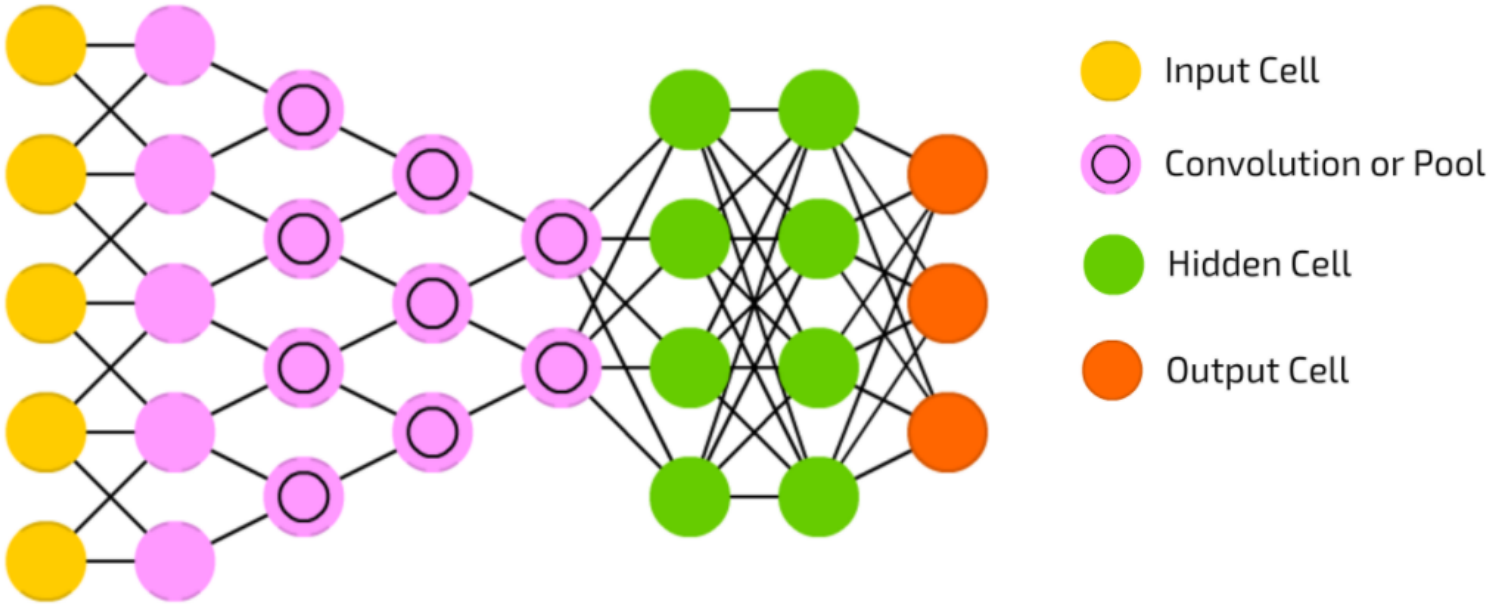
Recurrent Neural Network

- *Direct feedback* is the connection between the output and input of the same neuron.
- *Indirect feedback* is the connection between the output of a neuron and the input of a different neuron of the same layer.
- *Lateral feedback* is the connection between the output of a neuron and the input of a neuron of a previous layer.

Example: Text Recognition



Convolutional Neural Network (CNN)



- CNNs reduce the dimension of the input layer without loosing their characteristics.
- They use a hierarchical structure instead of fully connected cells.
- Then they build a neural net using the convolutions as input.

CNN Example: Tiger

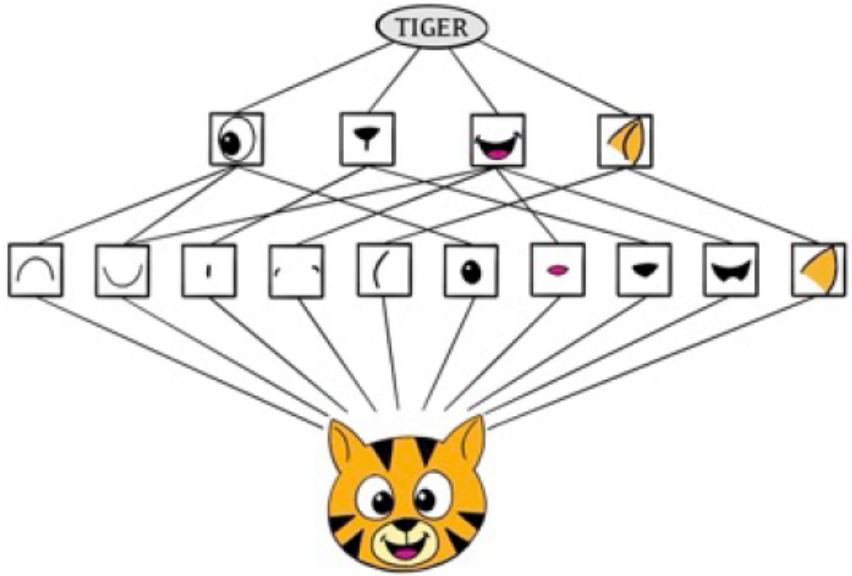


FIGURE 10.6. *Schematic showing how a convolutional neural network classifies an image of a tiger. The network takes in the image and identifies local features. It then combines the local features in order to create compound features, which in this example include eyes and ears. These compound features are used to output the label "tiger".*

- The network first identifies low-level features in the input image, such as small edges, patches of color, and the like.
- These low-level features are then combined to form higher-level features, such as parts of ears, eyes, and so on.

How does a CNN build up this hierarchy?

- CNN combine two specialized types of hidden layers, called convolution layers and pooling layers.
- Convolution layers search for instances of small patterns in the image, whereas pooling layers downsample these to select a prominent subset.
- In order to achieve state-of-the-art results, contemporary neuralnetwork architectures make use of many convolution and pooling layers. We describe convolution and pooling layers next.

Convolution Layers

A convolution layer is made up of a large number of convolution filters, each of which is a template that determines whether a particular local feature is present in an image.

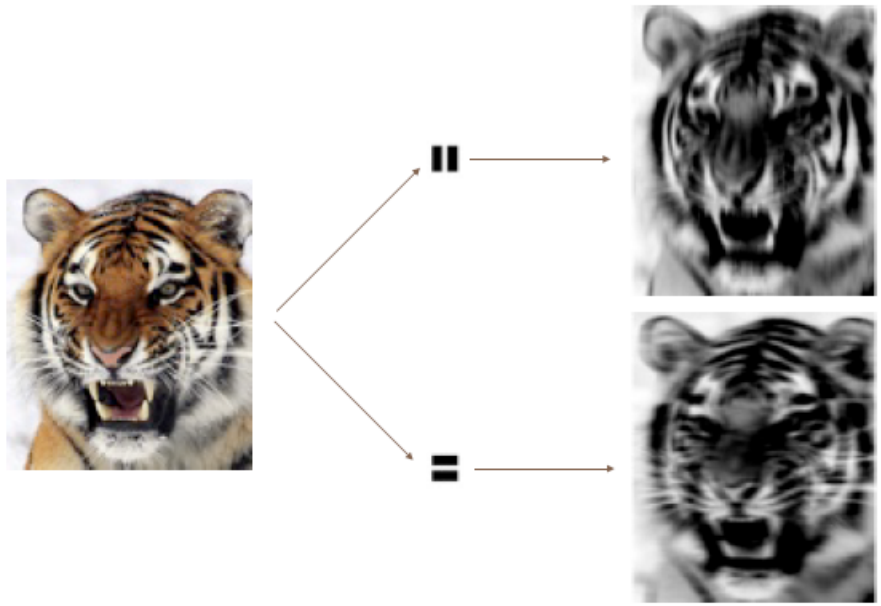


FIGURE 10.7. *Convolution filters find local features in an image, such as edges and small shapes. We begin with the image of the tiger shown on the left, and apply the two small convolution filters in the middle. The convolved images highlight areas in the original image where details similar to the filters are found. Specifically, the top convolved image highlights the tiger's vertical stripes, whereas the bottom convolved image highlights the tiger's horizontal stripes. We can think of the original image as the input layer in a convolutional neural network, and the convolved images as the units in the first hidden layer.*

Pooling Layers

A pooling layer provides a way to condense a large image into a smaller summary image.

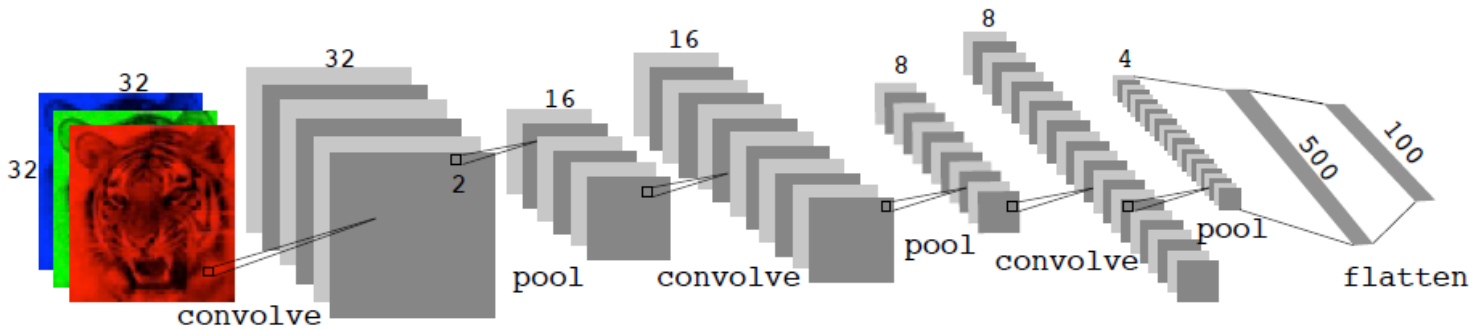
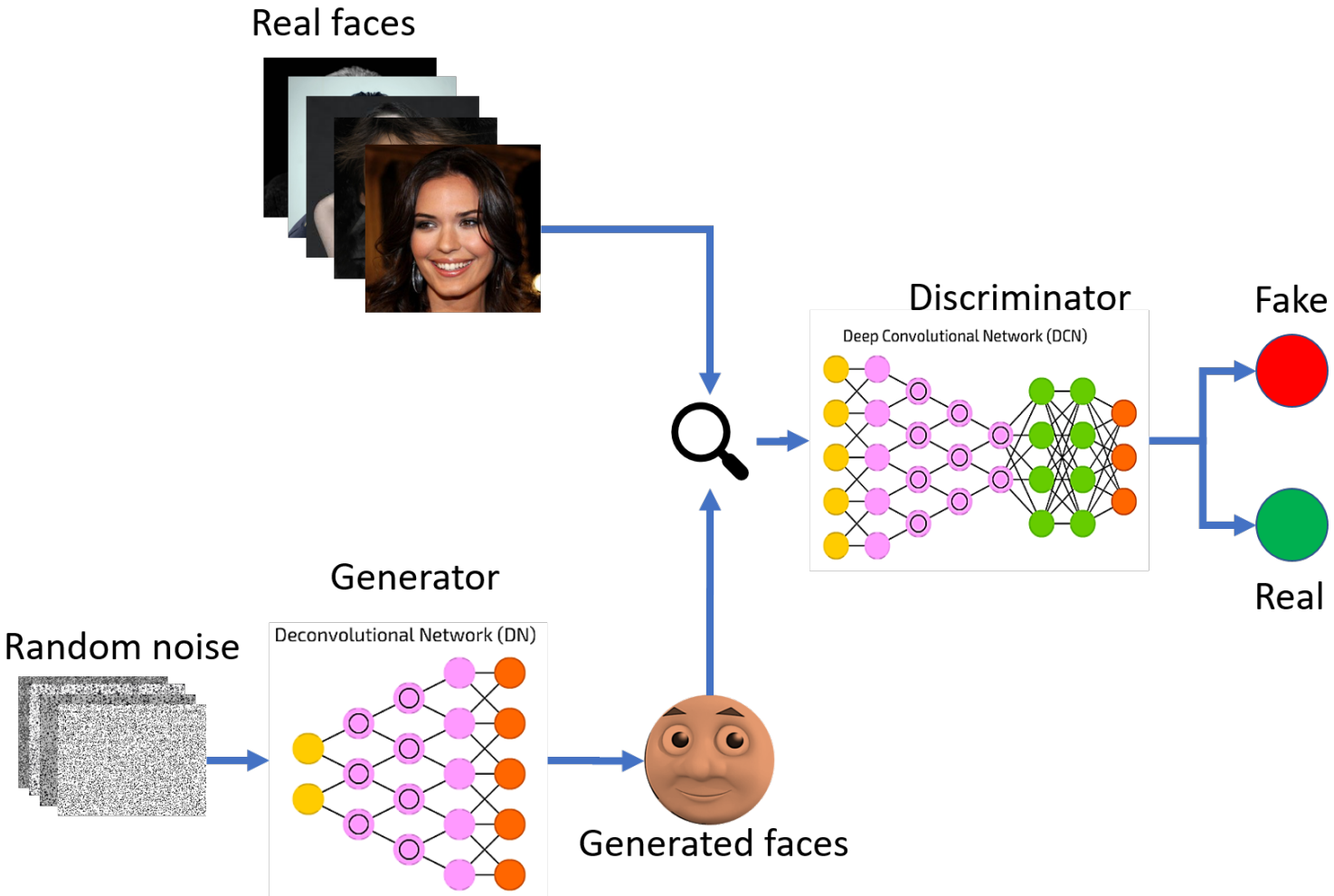
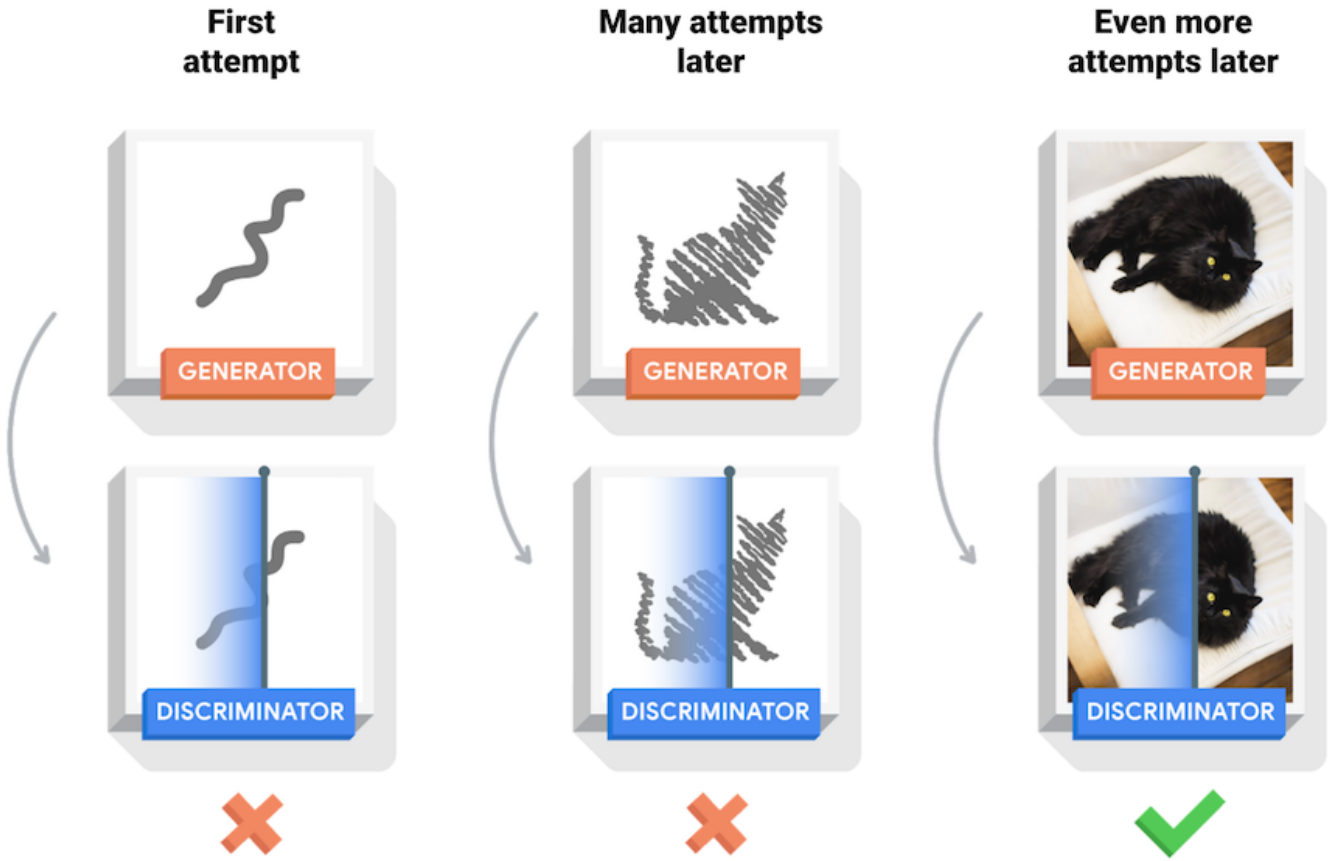


FIGURE 10.8. *Architecture of a deep CNN for the* `CIFAR100` *classification task. Convolution layers are interspersed with* 2×2 *max-pool layers, which reduce the size by a factor of 2 in both dimensions.*

Generative Adversarial Neural Networks (GANs)



Generative Adversarial Neural Networks (GANs)



When to Use Deep Learning?

- The performance of deep learning can be impressive for: image classification tasks, speech and language translation, forecasting, and text recognition
- *Should we discard all our older tools, and use deep learning on every problem with data?*
- Occam's razor principle: when faced with several methods that give roughly equivalent performance, pick the simplest.

Model	# Parameters	Mean Abs. Error	Test Set R^2
Linear Regression	20	254.7	0.56
Lasso	12	252.3	0.51
Neural Network	1345	257.4	0.54

TABLE 10.2. *Prediction results on the* `Hitters` *test data for linear models fit by ordinary least squares and lasso, compared to a neural network fit by stochastic gradient descent with dropout regularization.*

- Deep learning tends to be an attractive choice when the sample size is extremely large, the data structure is complex, and when interpretability of the model is not a high priority.