Session 1 Recap - DL Introduction and Applications

Nyandwi JD

Gedeon M

To cover

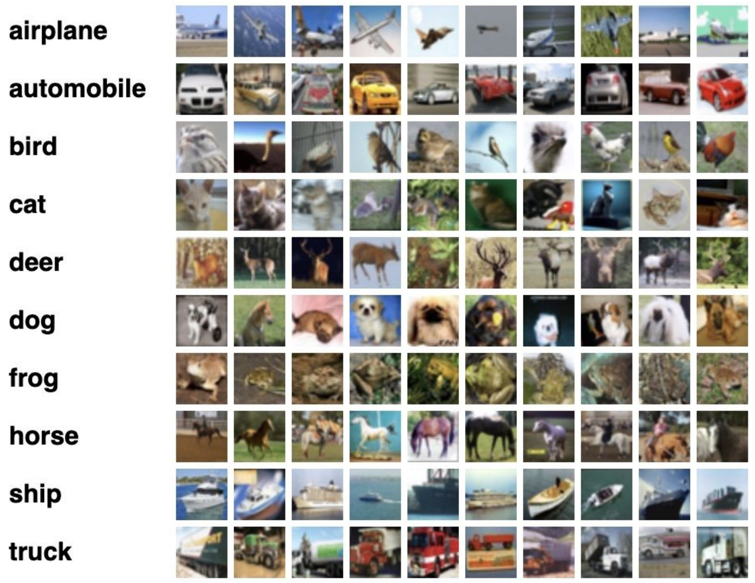
- Why and What is Deep Learning Again?
- Shallow Neural Network
- Multi-Layer Perceptrons
- Activation Functions
- Training Neural Networks: The intuition behind forward and backward pass
- Loss and Optimization Functions
- Overfitting Vs Underfitting
- Transfer learning

What is Deep Learning?

Deep learning is the study of artificial neural networks. It’s a subfield of machine learning that deals building systems that can extract meaningful features from large amount of data.

What is Deep Learning?

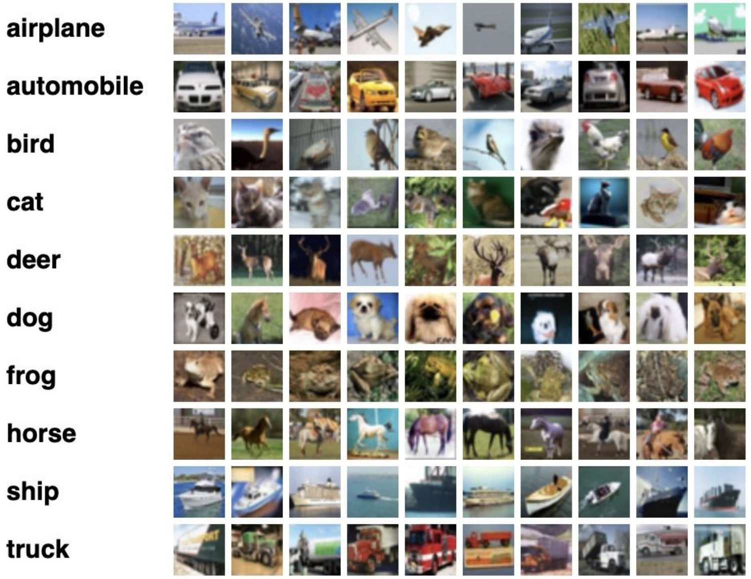
Deep learning is the study of artificial neural networks. It’s a subfield of machine learning that deals building systems that can extract meaningful features from large amount of data.



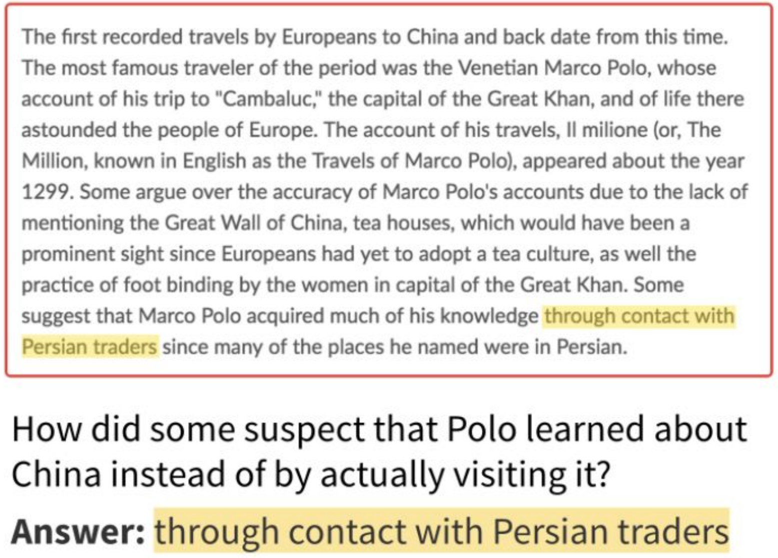
Cifar-100, Learning multiple layers of features from tiny images, Krizhevsky et al.

What is Deep Learning?

Deep learning is the study of artificial neural networks. It’s a subfield of machine learning that deals building systems that can extract meaningful features from large amount of data.



Cifar-100, Learning multiple layers of features from tiny images, Krizhevsky et al.



The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

QNLI (Question-answering NLI), Wang et al

Most large-scale problems requires extracting fine-grained features from the data. Different to humans who have inherent ability of recognizing things(images, speeches, texts), computers need to learn from massive amount of datasets.

Images: Pexels

Most large-scale problems requires extracting fine-grained features from the data. Different to humans who have inherent ability of recognizing things(images, speeches, texts), computers need to learn from massive amount of datasets.

When you see the following images, you know they are airplanes although they are all different in colors, shape, etc…

Images: Pexels

Most large-scale problems requires extracting fine-grained features from the data. Different to humans who have inherent ability of recognizing things(images, speeches, texts), computers need to learn from massive amount of datasets.

When you see the following images, you know they are airplanes although they are all different in colors, shape, etc…

For computers to recognize all above images are airplane, it's a whole different thing…

Images: Pexels

For computers to recognize that all above images are airplane, it’s a whole different thing. We have to train computers for that with many images of airplanes in different conditions.

Airplane

Images: Pexels

For computers to recognize that all above images are airplane, it’s a whole different thing. We have to train computers for that with many images of airplanes in different conditions.

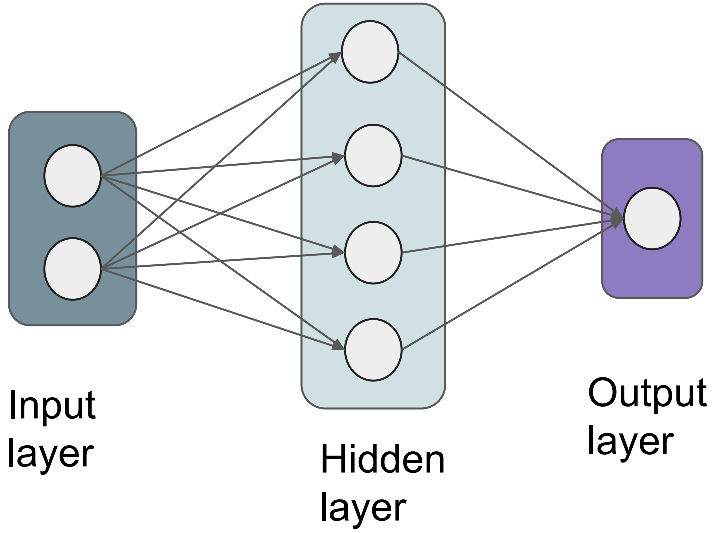
Airplane

Deep learning has shown potential in solving problems that seemed previously impossible. And the theme has always been the same: building a large deep learning models that can learn meaningful features from large amount of data.

Images: Pexels

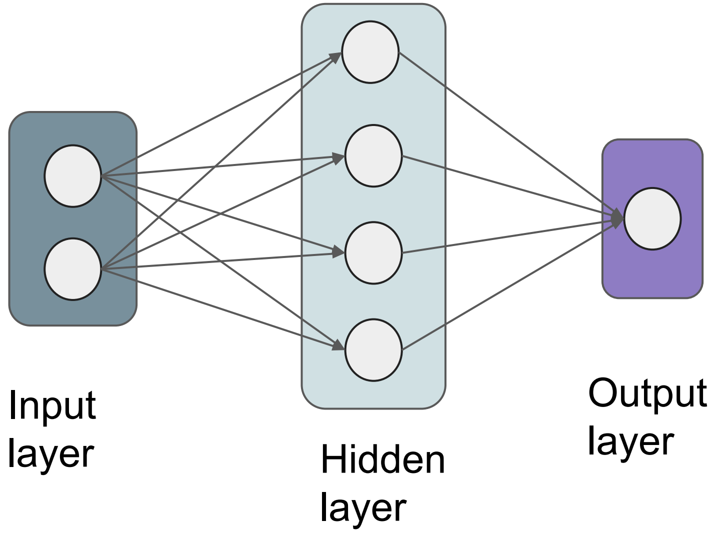
Shallow Neural Networks

Shallow neural networks typically have 3 layers: input layer, hidden layer, and output layer.



Shallow Neural Networks

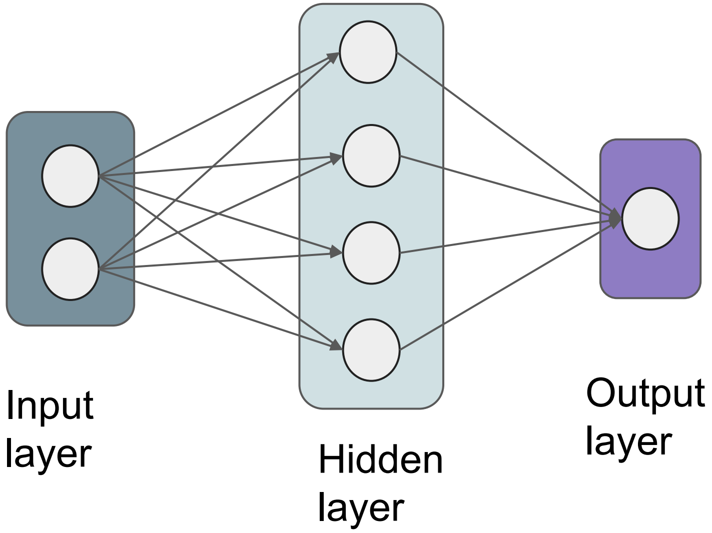
Shallow neural networks typically have 3 layers: input layer, hidden layer, and output layer.



- A shallow network is the simplest deep neural network.
- A layer is the building block of neural networks.
- The number of neurons in all layers depend on the problem. They are hyperparameters!
- If there more than 1 hidden layer, we tend to say deep neural network!

Shallow Neural Networks

Shallow neural networks typically have 3 layers: input layer, hidden layer, and output layer.



- A shallow network is the simplest deep neural network.
- A layer is the building block of neural networks.
- The number of neurons in all layers depend on the problem. They are hyperparameters!
- If there more than 1 hidden layer, we tend to say deep neural network!

Shallow networks is the name given according to the size of the network not the type of the architecture.

Multi-Layer Perceptrons(MLPs)

Multi-layer perceptrons is a stack of many linear layers. A linear layer is also called fully connected layer or dense layer. MLPs are also called densely connected neural networks.

Multi-Layer Perceptrons(MLPs)

Multi-layer perceptrons is a stack of many linear layers. A linear layer is also called fully connected layer or dense layer. MLPs are also called densely connected neural networks.

MLPs are simple multiple perceptrons. A perceptron was invented by Frank Rosenblatt in 1958. It is a binary classifier that can identify the class of a given input data.

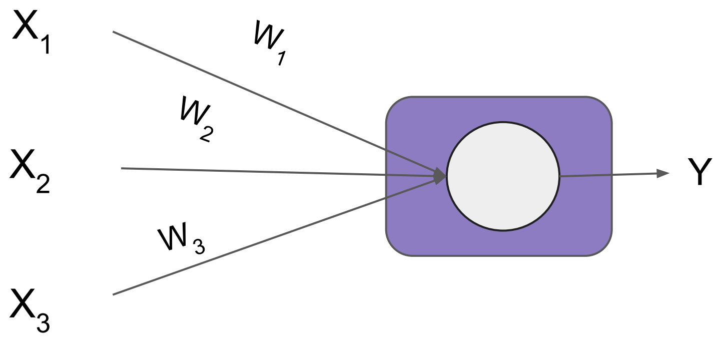
Multi-Layer Perceptrons(MLPs)

Multi-layer perceptrons is a stack of many linear layers. A linear layer is also called fully connected layer or dense layer. MLPs are also called densely connected neural networks.

MLPs are simple multiple perceptrons. A perceptron was invented by Frank Rosenblatt in 1958. It is a binary classifier that can identify the class of a given input data.

$$Y = \begin{cases} 1 \text{ if } W.X + B > 0 \\ 0 \text{ otherwise} \end{cases}$$

W: Weights
X: Input data
B: Biases



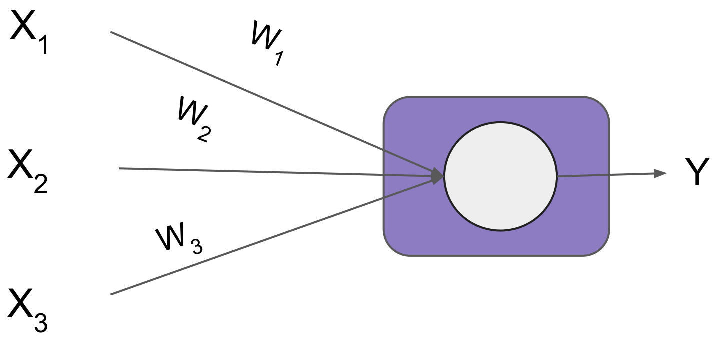
Multi-Layer Perceptrons(MLPs)

Multi-layer perceptrons is a stack of many linear layers. A linear layer is also called fully connected layer or dense layer. MLPs are also called densely connected neural networks.

MLPs are simple multiple perceptrons. A perceptron was invented by Frank Rosenblatt in 1958. It is a binary classifier that can identify the class of a given input data.

$$Y = \begin{cases} 1 \text{ if } W.X + B > 0 \\ 0 \text{ otherwise} \end{cases}$$

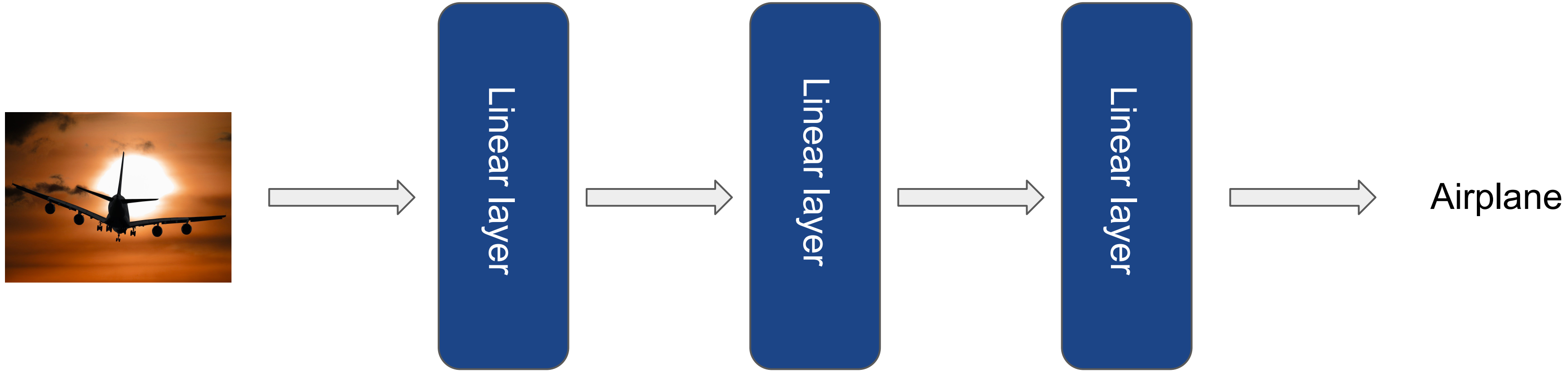
W: Weights
X: Input data
B: Biases



A linear layer in PyTorch

```
torch.nn.Linear(in_features,
out_features, bias=True)
```

Multi-Layer Perceptrons are simply stack of linear layers that goes from the input to output.

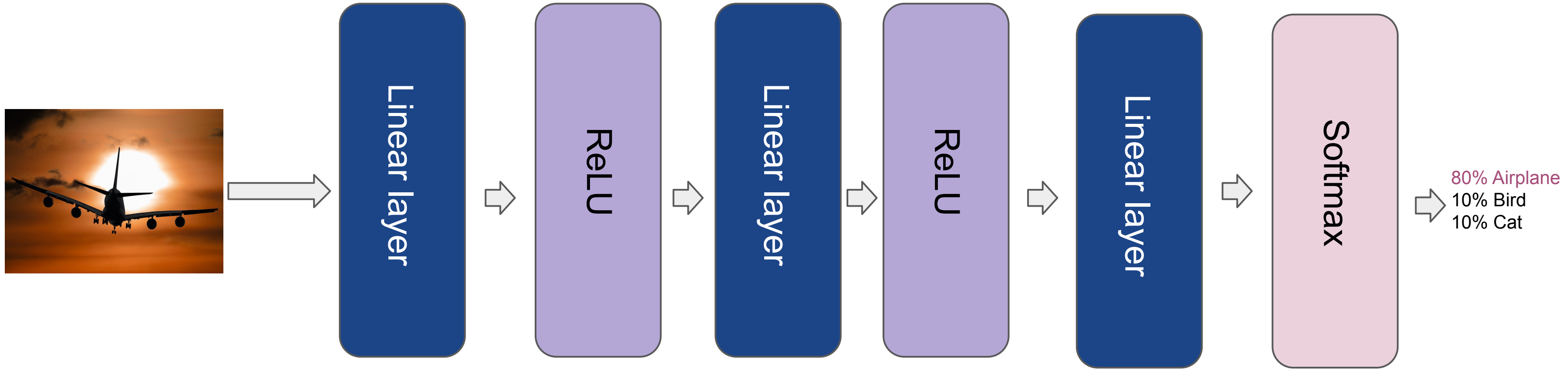


Activation Functions

Activation functions are non-linear functions that are used for introducing non-linearity into neural networks. Activation functions are usually inserted after linear layers(or other type of layers).

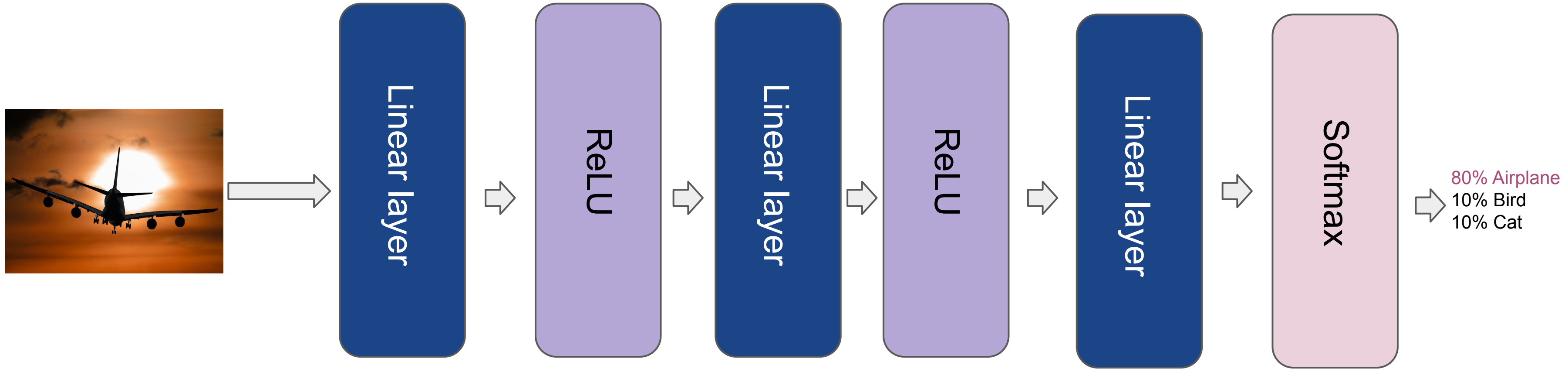
Activation Functions

Activation functions are non-linear functions that are used for introducing non-linearity into neural networks. Activation functions are usually inserted after linear layers(or other type of layers).



Activation Functions

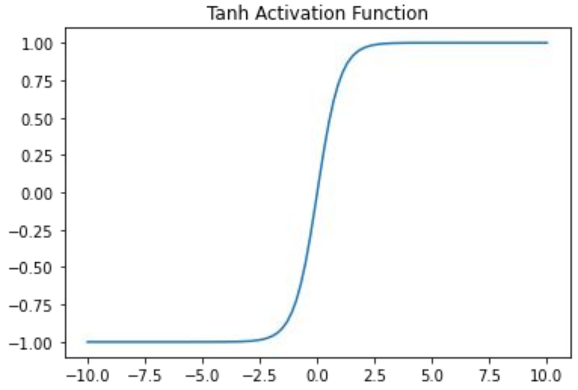
Activation functions are non-linear functions that are used for introducing non-linearity into neural networks. Activation functions are usually inserted after linear layers(or other type of layers).



ReLU or Rectified Linear Unit is the most used activation functions in hidden layer.

Commonly used activation functions

$$tanh(x)=2\sigma(2x)-1$$

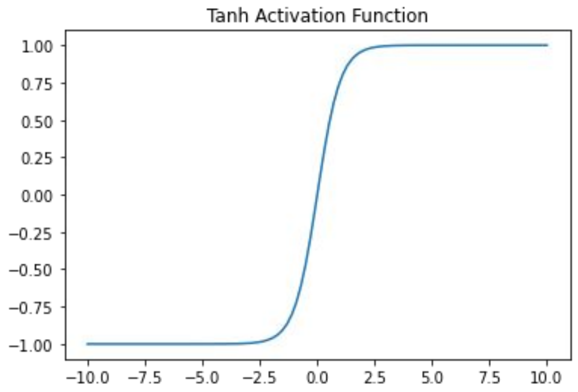


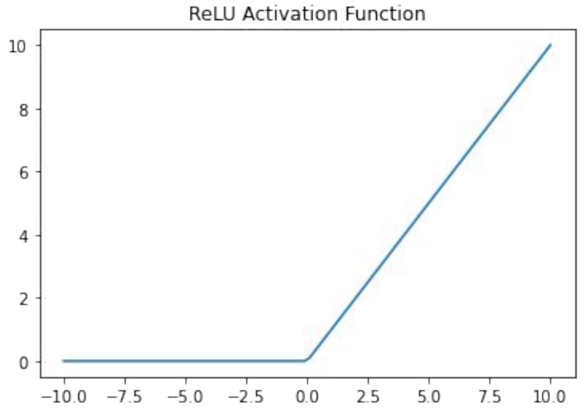
`nn.Tanh`

Commonly used activation functions

$$tanh(x) = 2\sigma(2x) - 1$$

$$ReLU(x) = max(0, x)$$





`nn.Tanh`

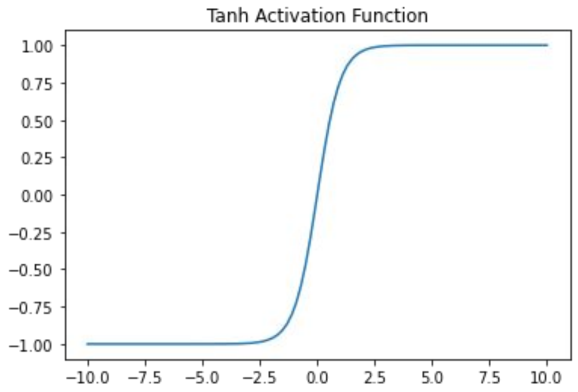
`nn.ReLU`

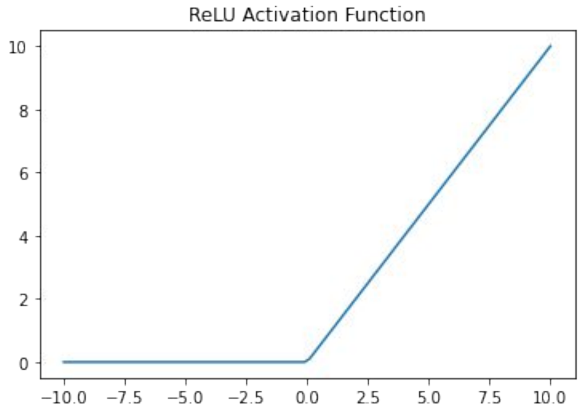
Commonly used activation functions

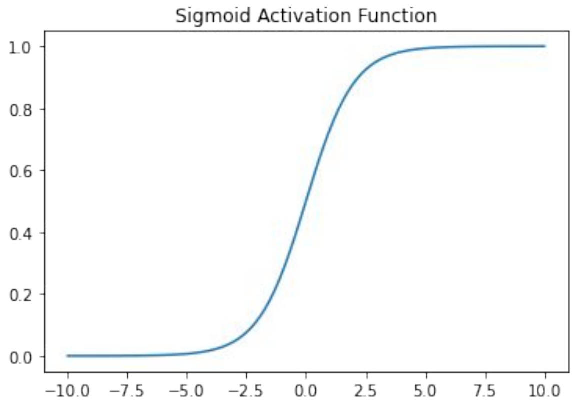
$$tanh(x)=2\sigma(2x)-1$$

$$ReLU(x)=max(0,x)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$







`nn.Tanh`

`nn.ReLU`

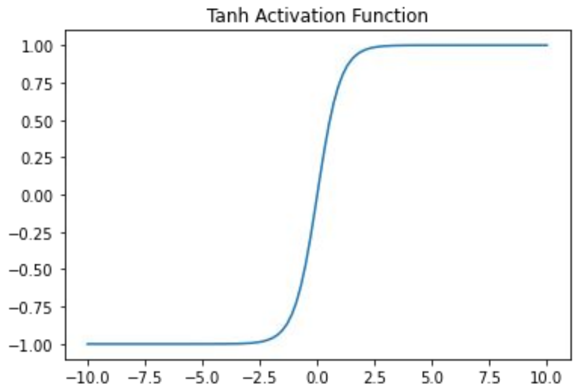
`nn.Sigmoid`

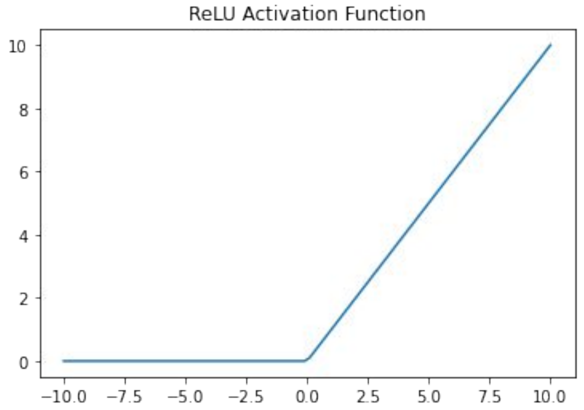
Commonly used activation functions

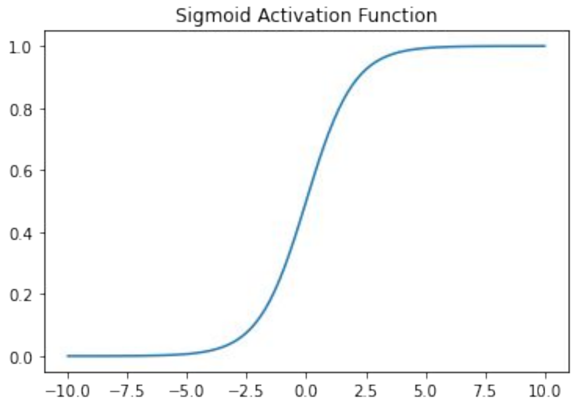
$tanh(x)=2\sigma(2x)-1$

$ReLU(x)=max(0,x)$

$\sigma(x) = \frac{1}{1+e^{-x}}$







`nn.Tanh`

`nn.ReLU`

`nn.Sigmoid`

Play with neural nets and activation functions: https://playground.tensorflow.org

How to choose activation functions:

- The rule of thumb is stick to ReLU, it works great for most problems and neural networks.

Image credit: Johnson

How to choose activation functions:

- The rule of thumb is stick to ReLU, it works great for most problems and neural networks.
- In classification tasks, the activation in output layer depends number of classes/labels. If you are doing binary classification, use sigmoid(with one neuron). If you have more than 2 classes, use softmax(with number of neurons equivalent to classes)

Image credit: Johnson

How to choose activation functions:

- The rule of thumb is stick to ReLU, it works great for most problems and neural networks.
- In classification tasks, the activation in output layer depends number of classes/labels. If you are doing binary classification, use sigmoid(with one neuron). If you have more than 2 classes, use softmax(with number of neurons equivalent to classes)

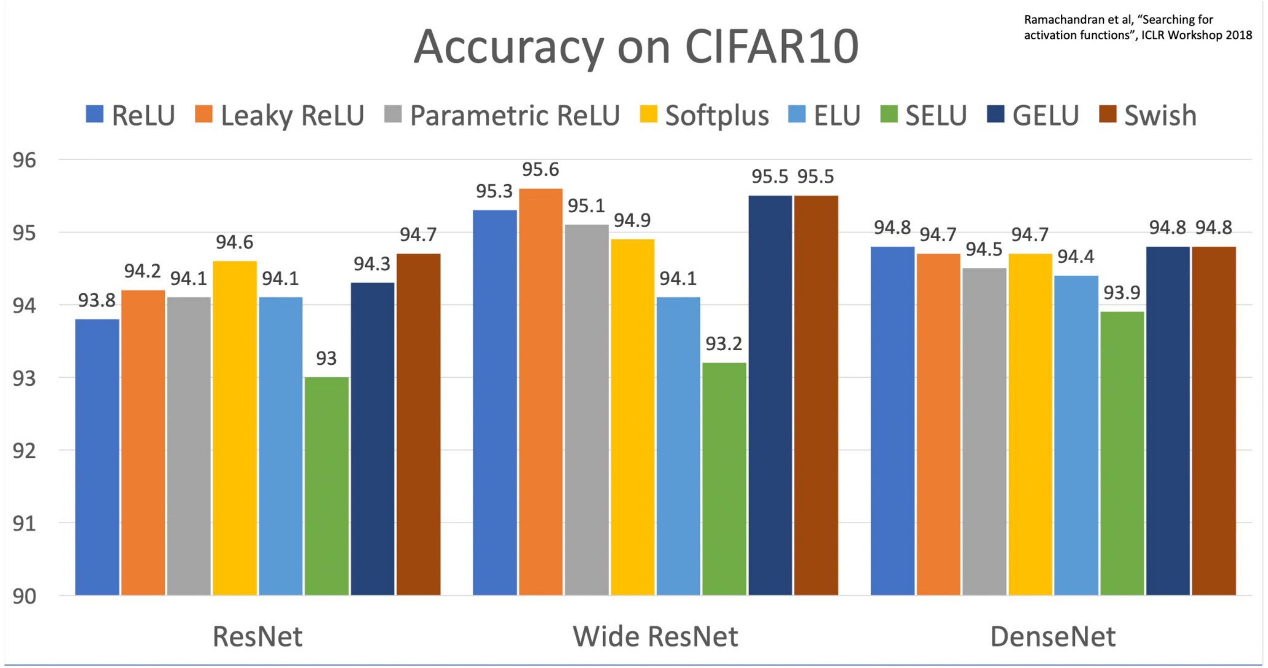


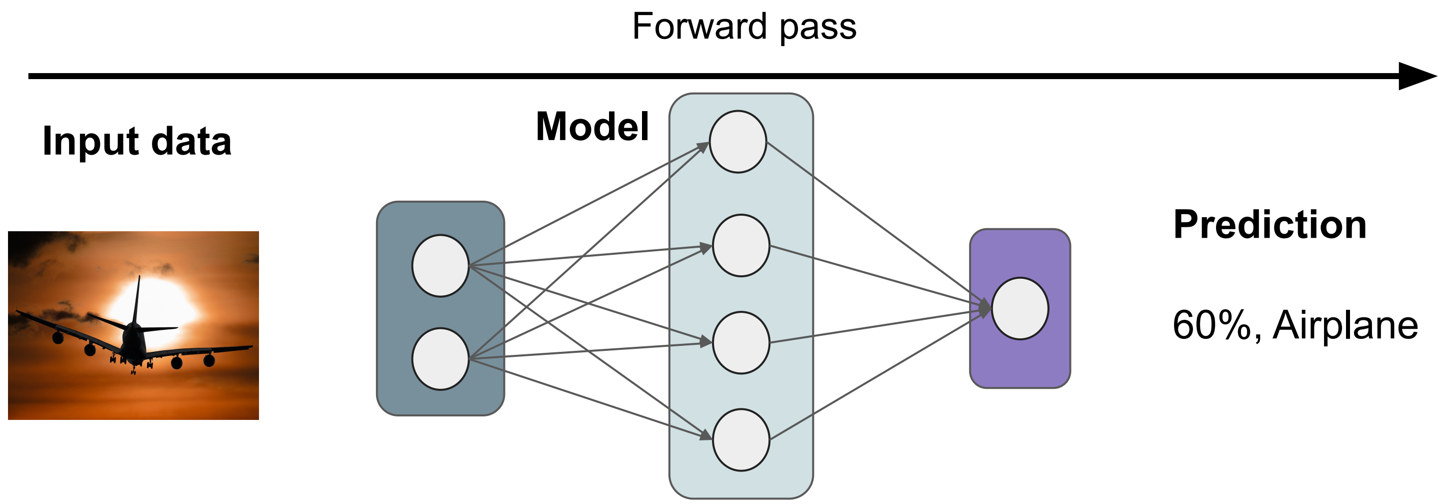
Image credit: Johnson

Loss Training Neural Networks: The high-level intuition behind forward and backward pass

Training neural networks is a two-way route and iterative process. In the first route, we feed the input data to the neural network and compute the prediction. This is forward pass or propagation.

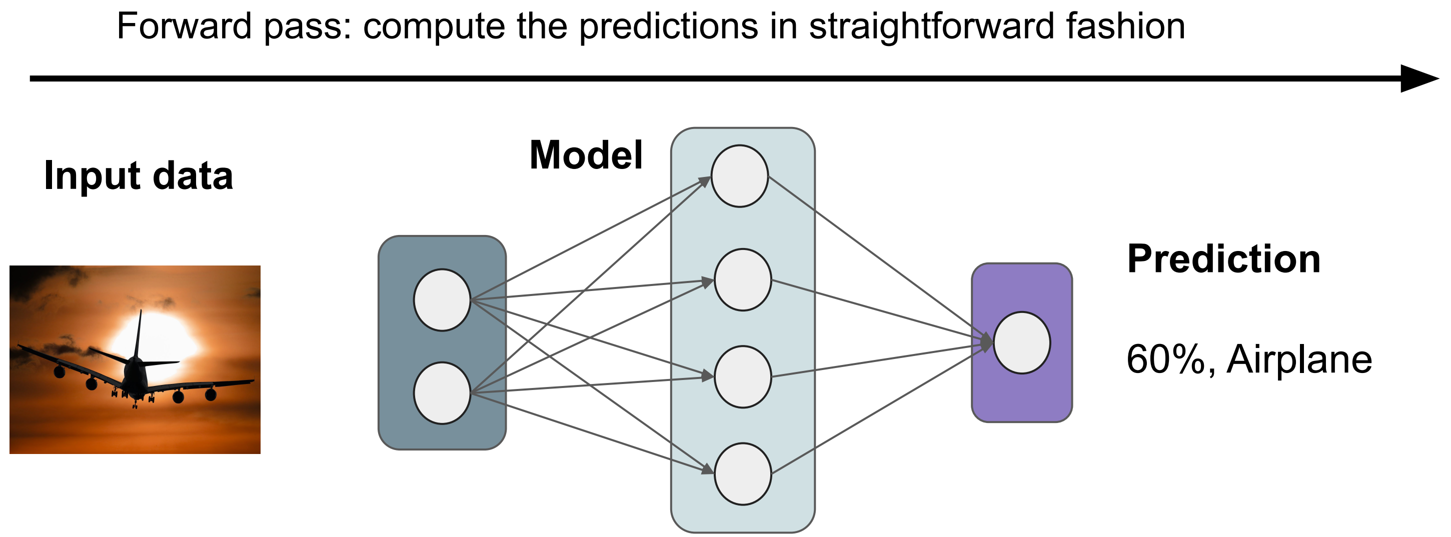
Loss Training Neural Networks: The high-level intuition behind forward and backward pass

Training neural networks is a two-way route and iterative process. In the first route, we feed the input data to the neural network and compute the prediction. This is forward pass or propagation.

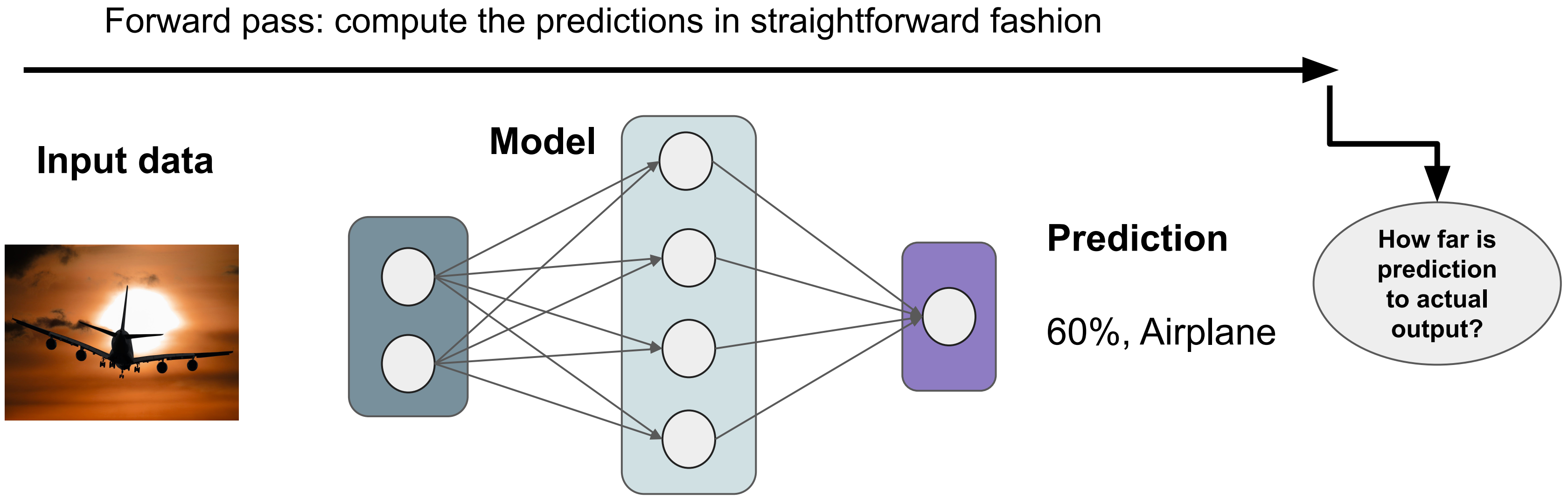


Very likely, in the first iterations, the predictions are far from the ground truth. So, we need to do something…..

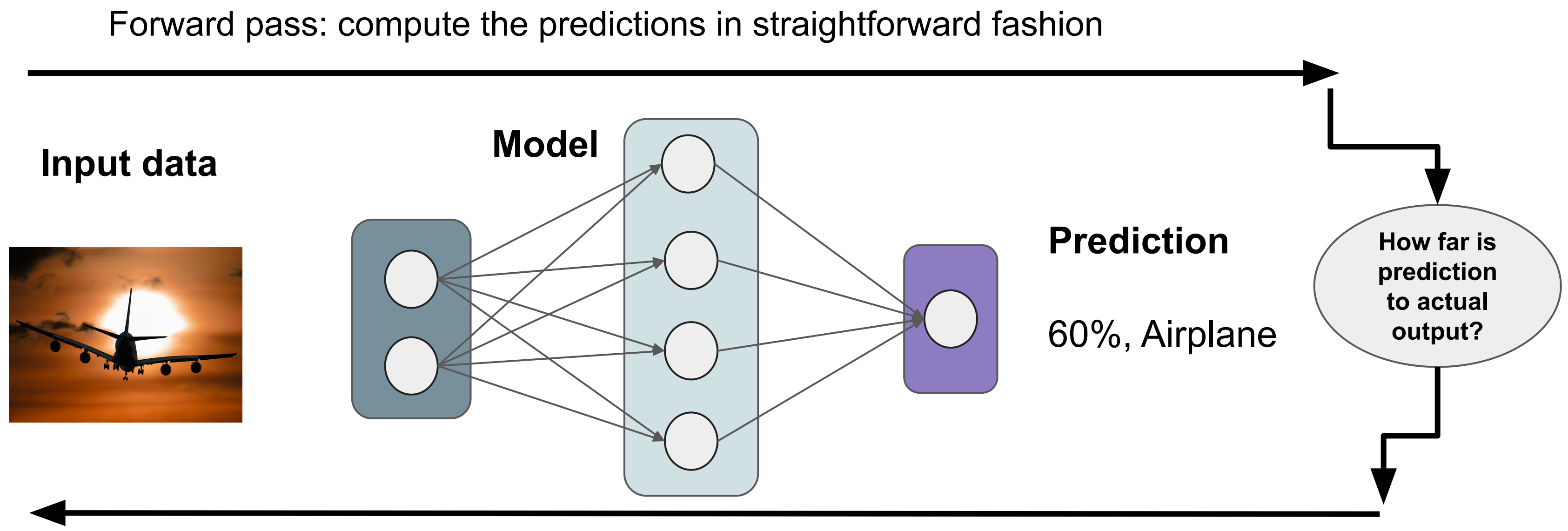
Very likely, in the first iterations, the predictions are far from the ground truth. To improve the prediction, we measure the difference between the predictions and ground truth(loss), and minimize such difference. The whole process of minimizing the loss is called backward propagation.


Forward pass: compute the predictions in straightforward fashion
Input data
Model
Prediction
60%, Airplane

Very likely, in the first iterations, the predictions are far from the ground truth. To improve the prediction, we measure the difference between the predictions and ground truth(loss), and minimize such difference. The whole process of minimizing the loss is called backward propagation.


Forward pass: compute the predictions in straightforward fashion
Input data
Model
Prediction
60%, Airplane
How far is prediction to actual output?

Very likely, in the first iterations, the predictions are far from the ground truth. To improve the prediction, we measure the difference between the predictions and ground truth(loss), and minimize such difference. The whole process of minimizing the loss is called backward propagation.


Forward pass: compute the predictions in straightforward fashion
Input data
Model
Prediction
60%, Airplane
How far is prediction to actual output?

Backward pass: measure the difference between prediction and ground truth, and minimize it with gradient optimization techniques.

Loss and Optimization Functions

As we said previously, training neural network is iterative process. At every training iteration, we compute the predictions, measure the difference between predictions and output with loss functions, and minimize the loss with various optimization techniques.

Examples of loss functions by tasks:

- Regression tasks: mean squared error(MSE), mean absolute error(MAE), etc…
- Classification: binary cross-entropy, categorical cross entropy, sparse categorical cross entropy, etc…

Loss and Optimization Functions

As we said previously, training neural network is iterative process. At every training iteration, we compute the predictions, measure the difference between predictions and output with loss functions, and minimize the loss with various optimization techniques.

Examples of loss functions by tasks:

- Regression tasks: mean squared error(MSE), mean absolute error(MAE), etc…
- Classification: binary cross-entropy, categorical cross entropy, sparse categorical cross entropy, etc…

Optimization functions are used for minimizing the loss functions. The commonly used activations are adam, stochastic gradient descent(SGD), RMSprop, Nadam etc…

Loss and Optimization Functions

As we said previously, training neural network is iterative process. At every training iteration, we compute the predictions, measure the difference between predictions and output with loss functions, and minimize the loss with various optimization techniques.

Examples of loss functions by tasks:

- Regression tasks: mean squared error(MSE), mean absolute error(MAE), etc…
- Classification: binary cross-entropy, categorical cross entropy, sparse categorical cross entropy, etc…

Optimization functions are used for minimizing the loss functions. The commonly used activations are adam, stochastic gradient descent(SGD), RMSprop, Nadam etc…

Loss functions are very specific to tasks, they are easier to choose. Optimizers depends on many variables. As a rule of thumb, don’t be a hero. Stick to Adam!!

Overfitting vs Underfitting

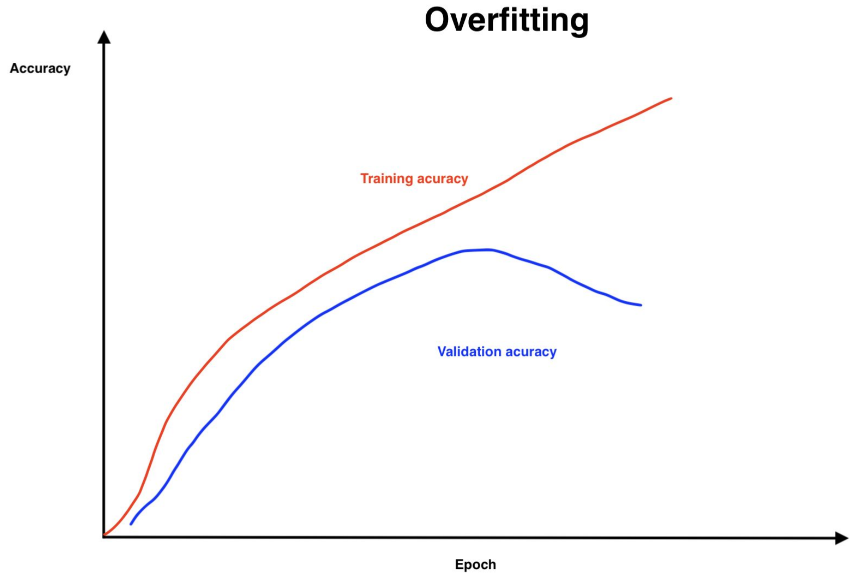
Overfitting and underfitting are the two main challenges of training neural networks. Overfitted model will perform well on training data but not so well on validation data.

Overfitting is usually caused by using large model, small training data, and data leakage.

Overfitting vs Underfitting

Overfitting and underfitting are the two main challenges of training neural networks. Overfitted model will perform well on training data but not so well on validation data.

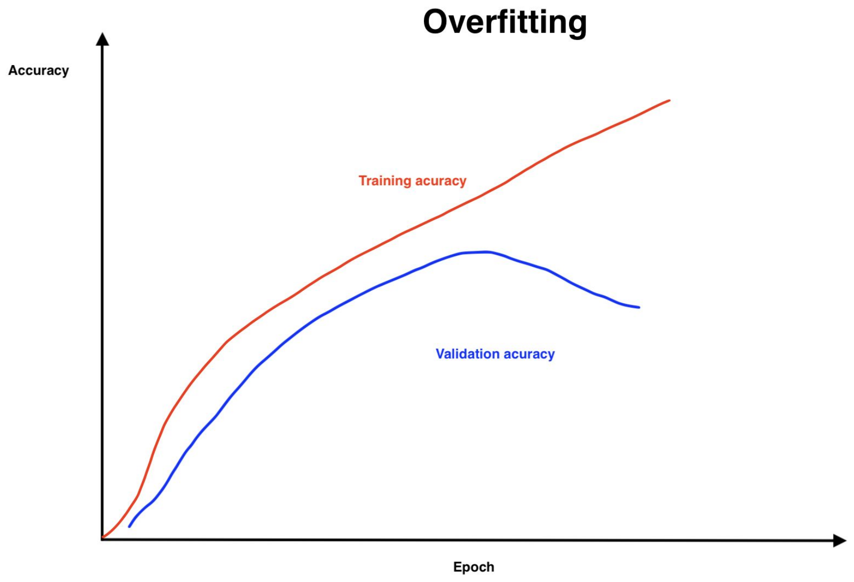
Overfitting is usually caused by using large model, small training data, and data leakage.



Overfitting vs Underfitting

Overfitting and underfitting are the two main challenges of training neural networks. Overfitted model will perform well on training data but not so well on validation data.

Overfitting is usually caused by using large model, small training data, and data leakage.



Overfitting can be minimized by using:
- Increasing training data
- Data augmentation/using artificial data
- Early stopping
- Dropout
- Reducing the size of the model

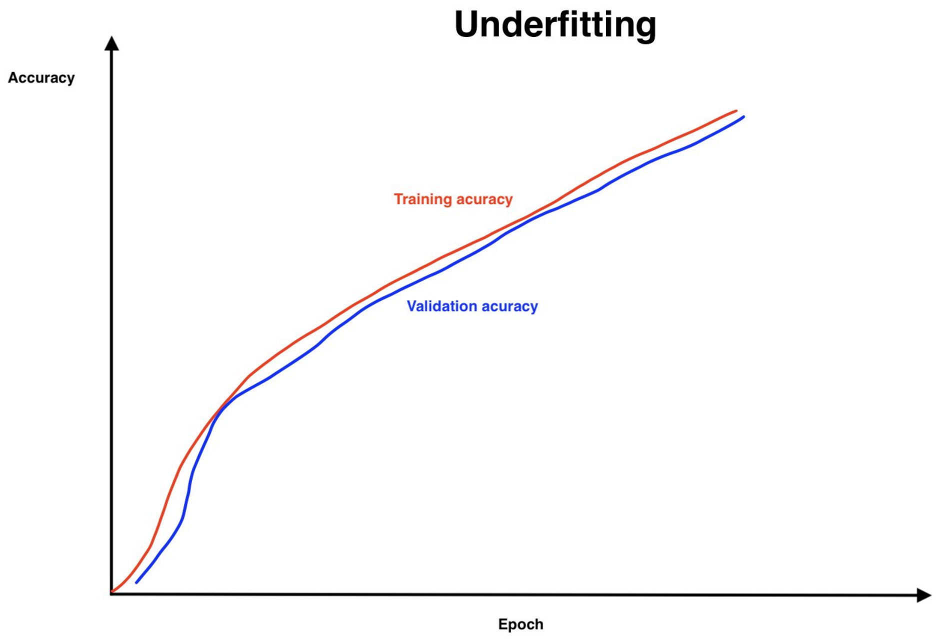
Data augmentation, early stopping, dropout are examples of regularization techniques.

Neural networks can also underfit. An under-fitted model will look like its doing well on both training data and validation data, but it’s actually not the case. Also, an under-fitted model can have a poor performance on training data. The later is the typical explanation of underfitting.

Underfitting is caused by using a small model, poor data(data with low predictive power), etc…

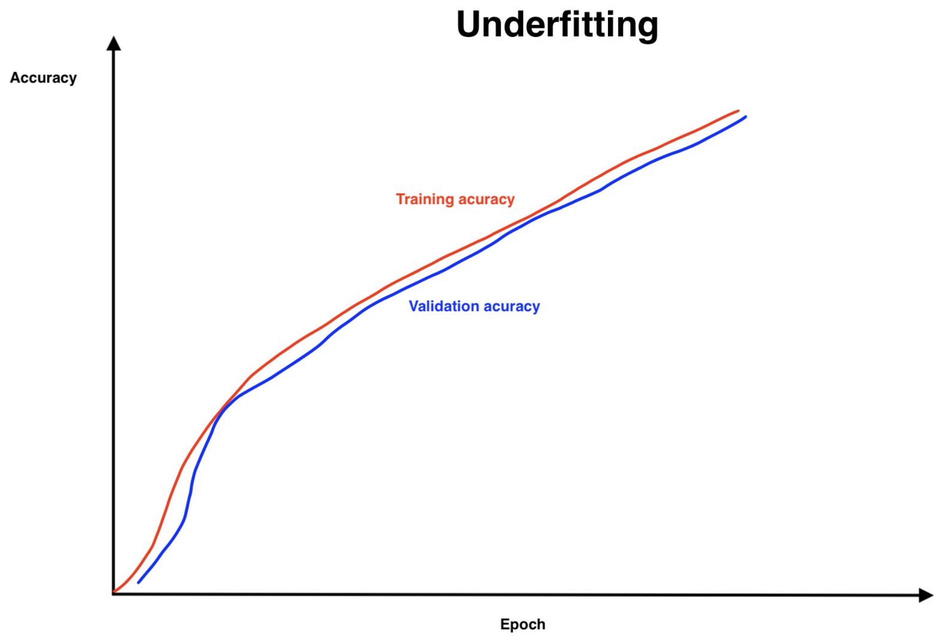
Neural networks can also underfit. An under-fitted model will look like its doing well on both training data and validation data, but it’s actually not the case. Also, an under-fitted model can have a poor performance on training data. The later is the typical explanation of underfitting.

Underfitting is caused by using a small model, poor data(data with low predictive power), etc…



Neural networks can also underfit. An under-fitted model will look like its doing well on both training data and validation data, but it’s actually not the case. Also, an under-fitted model can have a poor performance on training data. The later is the typical explanation of underfitting.

Underfitting is caused by using a small model, poor data(data with low predictive power), etc…



Underfitting can be handled by:

- Trying large model
- Improving the data quality

Transfer learning

A popular notion in deep learning is that no one should be building and training their neural networks from scratch! It’s a very high price to pay!

Transfer learning

A popular notion in deep learning is that no one should be building and training their neural networks from scratch! It’s a very high price to pay!

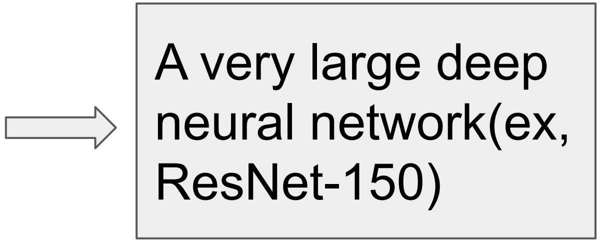
What people tend to do is to re-use a large pre-trained model(vision or language model) or adapt to their own problems.

Transfer learning in computer vision

Train a large model on very large dataset(typically ImageNet) and reus-use the trained model on a new small dataset.

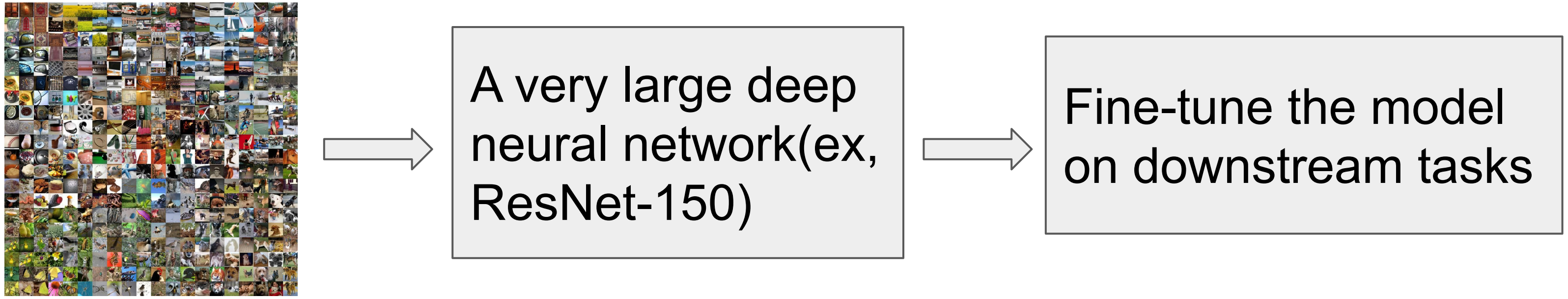
Transfer learning in computer vision

Train a large model on very large dataset(typically ImageNet) and reus-use the trained model on a new small dataset.



Transfer learning in computer vision

Train a large model on very large dataset(typically ImageNet) and reus-use the trained model on a new small dataset.


A very large deep neural network(ex, ResNet-150)
Fine-tune the model on downstream tasks

Example of computer vision downstream tasks: image classification, object detection, segmentation

Transfer learning in computer vision

Train a large model on very large dataset(typically ImageNet) and reus-use the trained model on a new small dataset.

$\Rightarrow$

A very large deep neural network(ex, ResNet-150)

$\Rightarrow$

Fine-tune the model on downstream tasks

Big research labs train large models and open-source them. Most people should only using them…

Example of computer vision downstream tasks: image classification, object detection, segmentation

Transfer learning in NLP

Train a large language model(nowadays transformers) on a massive text dataset and use the pre-trained model on small dataset.

WIKIPEDIA
The Free Encyclopedia

English
6 458 000+ articles

日本語
1 314 000+ 記事

Español
1 755 000+ artículos

Русский
1 798 000+ статей

Deutsch
2 667 000+ Artikel

Français
2 400 000+ articles

Italiano
1 742 000+ voci

中文
1 256 000+ 条目 / 條目

Português
1 085 000+ artigos

العربية
+000 159 1 مقالة

Transfer learning in NLP

Train a large language model(nowadays transformers) on a massive text dataset and use the pre-trained model on small dataset.

$\Rightarrow$

A very large language model(ex: BERT)

Example of NLP downstream tasks: text classification, sentiment analysis, question answering, etc…

Transfer learning in NLP

Train a large language model(nowadays transformers) on a massive text dataset and use the pre-trained model on small dataset.

→ A very large language model(ex: BERT) → Fine-tune the model on downstream tasks

Example of NLP downstream tasks: text classification, sentiment analysis, question answering, etc…

End…What we covered

- Why and What is Deep Learning Again?
- Shallow Neural Network
- Multi-Layer Perceptrons
- Activation Functions
- Training Neural Networks: The intuition behind forward and backward pass
- Loss and Optimization Functions
- Overfitting Vs Underfitting
- Transfer learning

Questions & Comments?

https://bit.ly/MLPs-Practice