About this workshop

In this workshop on TensorFlow, you'll dive into the fundamentals of building neural networks using the powerful tools provided by TensorFlow. We'll explore the essentials, including understanding the architecture of a neural network through dense layers, the impact of activation functions on model behavior, the role of optimizers in improving training efficiency, and how to measure and enhance model performance using various loss functions.

Additionally, we'll delve into the practical aspect of visualizing your model's insights through Matplotlib graphs, gaining a hands-on experience that bridges theory and application. By the end of the workshop, you'll have a solid understanding of key concepts in deep learning, empowering you to create and optimize neural networks for various tasks.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

Why do we use np.array and not python list?

Main reason: **performance** (take lesser memory and give faster execution, therefore resulting in faster operations)

What is a sequential class?

The Sequential class allows you to create models layer by layer in a step-by-step fashion.

Advantages:

The Sequential class provides a clean and simple syntax for building and compiling models, making it suitable for beginners and for quick prototyping.

How to decide number of units

Input Layer: The number of neurons in the input layer is determined by the **number of features (elements) in your dataset**. Each neuron corresponds to one feature.

Output Layer: The number of neurons in the output layer **depends on the nature of your task:**

For **binary classification**, use 1 neuron with a sigmoid activation function.

For **multi-class classification**, use as many neurons as there are classes with a softmax activation function.

For **regression** tasks, use 1 neuron without an activation function or with a linear activation function.

Hidden Layers: Deciding the number of neurons in hidden layers **is more art than science**. Here are some strategies:

Rule of Thumb: A common rule of thumb is to use a number of neurons between the number of input and output neurons. Experiment with different sizes based on the complexity of your problem.

Start Small: Begin with a smaller number of neurons and gradually increase the size if needed. Overly complex architectures can lead to overfitting.

What is ReLU?

In mathematics form:

ReLU is a simple function: f(x) = max(0, x). It takes an input *x*, and if *x* is negative, it outputs zero; otherwise, it outputs *x*.

What is x?

In a neural network, each connection between neurons has a weight associated with it. The input *x* is the **weighted sum of the inputs to a neuron**, often combined with a bias term. Mathematically, it can be represented as:

$$x = \sum_i (w_i \cdot a_i) + b$$

- w_i represents the weights of the connections.
- a_i represents the inputs to the neuron.
- b is the bias term.

What is Sigmoid?

In mathematics form:

It transforms any real-valued number into a value between 0 and 1.

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

- e is the base of the natural logarithm (approximately equal to 2.71828).
- x is the input to the function.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is TanH?

- It transforms any real-valued number into a value between -1 and 1. The tanh function is defined mathematically as follows:

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

- e is the base of the natural logarithm (approximately equal to 2.71828).
- x is the input to the function.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is linear?

- the output of the linear activation is equal to the input itself. It does not introduce any non-linearity to the network, and its purpose is to provide a simple scaling of the input. While it's not commonly used in hidden layers for deep neural networks, it is sometimes used in specific scenarios, such as regression problems, where the output needs to be directly proportional to the input.

$$f(x) = x$$

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is softmax?

- It transforms a vector of real numbers into a probability distribution.

Given an input vector z with K elements $(z_1, z_2, \ldots, z_K)$, the softmax activation function computes the i-th element of the output vector $\sigma(z)$ as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Here's a breakdown of the components:

- $\sigma(z)_i$ is the probability assigned to the i-th element in the output vector.
- e is the base of the natural logarithm (approximately equal to 2.71828).
- z_i is the i-th element of the input vector.
- $\sum_{j=1}^{K} e^{z_j}$ is the sum of the exponentials of all elements in the input vector.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is “i” and what is “K” ?

You

lets say i have 5 classes, car, bus, airplane, bike, cycle now what is i and what is k

ChatGPT

In your example:

- K is the total number of classes or categories, which is 5 in this case. So, $K = 5$ because you have 5 classes: car, bus, airplane, bike, and cycle.
- i represents the index of a specific class within the range of 1 to K. For example:
  - If you are calculating the probability for the class "car," then $i = 1$.
  - If you are calculating the probability for the class "bus," then $i = 2$.
  - And so on.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is optimizer?

- Imagine you're trying to teach a computer to recognize handwritten digits. The optimizer's role is to fine-tune the model's settings during training so that it gets better and better at making accurate predictions.

Explain SGD

- Imagine you're on a treasure hunt in a big, hilly area. Your goal is to find the lowest point, where the treasure is buried. You're blindfolded, and you can only figure out if you're going uphill or downhill by feeling the ground.
- Now, you want to reach the lowest point (the treasure) as quickly as possible. For each step, you take a small jump and feel if the ground is going up or down. If you feel you're going downhill, you continue in that direction; if you feel you're going uphill, you change your direction.
- This jumping and feeling process is like what SGD does in machine learning. Instead of looking at the entire area (all the data) at once, SGD looks at a small part (a mini-treasure hunt) and adjusts its direction based on that. It keeps doing this until it finds the lowest point (the best settings for a problem).

Explain Adam

- Adam is like an assistant that not only helps the computer learn but also keeps track of how quickly it's learning and adjusts the learning process accordingly.
- **Learning Rate Control:**
- It means it automatically adapts the size of the steps the computer takes during learning. If the computer is learning quickly, it takes smaller steps; if it's learning slowly then it takes bigger steps.
- **Momentum:**
- It means to keep going in the same direction if learning process is being consistently improved.

RMSProp features

1. **Adaptive Learning Rate:**
   1. RMSprop adapts the learning rate individually for each parameter by dividing the learning rate by the square root of the average of the squared gradients. This helps to normalize the learning rate based on the historical gradients.
2. **Exponential Moving Average:**
   1. This moving average gives more weight to recent gradients and less weight to older ones, making the optimization process more adaptive to changing conditions.
3. **Bias Correction:**
   1. To account for bias in the early stages of training when the number of updates is small, RMSprop introduces bias correction. It adjusts the moving average by dividing it by a term that increases with the number of iterations.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

Binary crossentropy

- It measures the difference between the actual outcomes and the predictions made by a model for binary classification tasks.

The formula for Binary Crossentropy is as follows:

$$L(y, \hat{y}) = -\left(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right)$$

Here:

- $L(y, \hat{y})$ is the binary crossentropy loss.
- y is the actual class label (0 or 1).
- $\hat{y}$ is the predicted probability that the instance belongs to class 1.

In words, the formula says that for each instance:

- If $y = 1$ (actual class is positive), it penalizes the model more when the predicted probability $\hat{y}$ is closer to 0.
- If $y = 0$ (actual class is negative), it penalizes the model more when the predicted probability $\hat{y}$ is closer to 1.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

Difference between sparse and categorical crossentropy

Input for labels in sparse categorical crossentropy = [5]

Input for labels in categorical crossentropy = [0,0,0,0,0,1,0,0,0,0]

Categorical Crossentropy:

Imagine you have a robot friend who's really good at recognizing fruits. For every fruit it sees, it not only tells you what it thinks the **fruit is but also how confident it is about that guess. If you show it an apple, it might say, "I'm 90% sure it's an apple, 5% sure it's an orange, and 5% sure it's a banana."** Categorical crossentropy is like a tool that measures how well your robot friend's guesses match the actual fruits and how confident it is in its answers.

Sparse Categorical Crossentropy:

Now, let's say you have another friend who's not as detailed. This friend can still recognize fruits, but when you show them an apple, they just say, "**Yep, that's an apple**." They don't give you a detailed breakdown of confidence like the robot does. Sparse categorical crossentropy is like a tool for this friend. It only cares about whether your friend can correctly say, "apple" or "orange" without worrying about the detailed confidence scores.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/

What is validation data?

- It is a separate set of data that the model does not see during the training process. After each training epoch (iteration), the model's performance is evaluated on the validation data. This evaluation helps us in understanding how the model is performing on unseen data.

- We generally use validation data to prevent overfitting (i.e. when model performs well on training data but not on new data)

What are feed forward neural networks?

A Feedforward Neural Network (FNN), also known as a Multi-Layer Perceptron (MLP), is a type of artificial neural network where **information moves in one direction**—forward—from the input layer, through hidden layers, and finally to the output layer. **There are no loops.**

Where are feedforwards neural networks used?

Feedforward Neural Networks (FNNs) are used in various applications across different domains due to their versatility and effectiveness in capturing complex patterns.

Image pattern recognition, natural language processing (text classification) , speech recognition, games (to create ai bots)

What is regression

Imagine you have a friend who loves playing video games. Let's call your friend Sam. Now, Sam noticed that the more time they spend practicing math problems, the better their scores get.

Regression is a bit like figuring out how much Sam's math scores improve based on how much time they spend practicing. So, if Sam practices for one hour, their score might go up by 10 points. If they practice for two hours, it might go up by 20 points. The goal is to understand this relationship and predict how much their score will improve for any amount of practice time.

LinkedIn: https://www.linkedin.com/in/abdulla-shahzan-03b00a226/