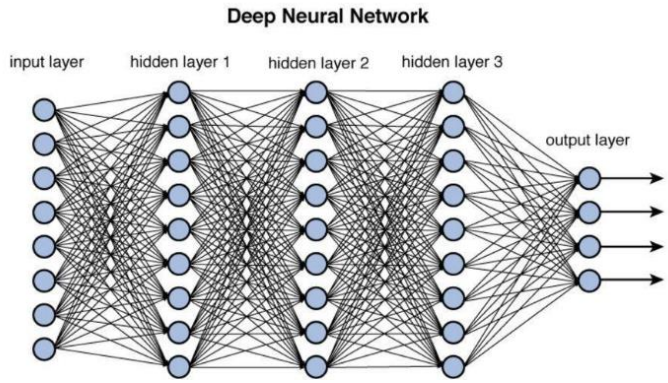
Week 8 Kahoot

Q1



Each input node corresponds to

- one batch of data points
- one data point
- **one input feature**
- one weight of the network

Q2

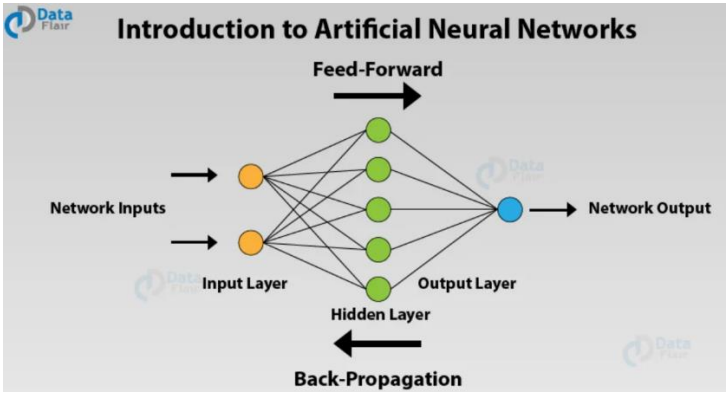
```
model= nn.Sequential(
            nn.Linear(2, 3),
            nn.ReLU(),
            nn.Linear(3, 1)
        )

summary(model, (1, 2))
```

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Sequential                               [1, 1]                    --
├─Linear: 1-1                            [1, 3]                    X
├─ReLU: 1-2                              [1, 3]                    --
├─Linear: 1-3                            [1, 1]                    4
==========================================================================
```

NN has 2 neurons in input layer, 3 in hidden and 1 in output. What is X on the image (number of parameters)

- 2
- 3
- 6
- **9**

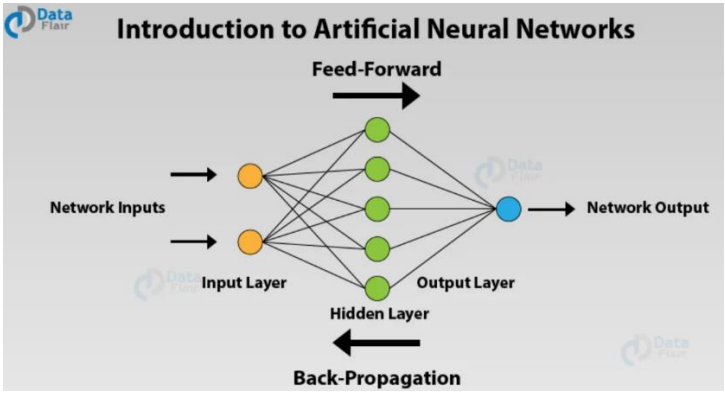
Q3



What is the role of a loss function in training neural networks?

- To define network architecture
- To control the training process by adjusting the learning rate
- **To quantify how well the model's predictions match the actual target values**
- To maximize the accuracy of the model on the training data

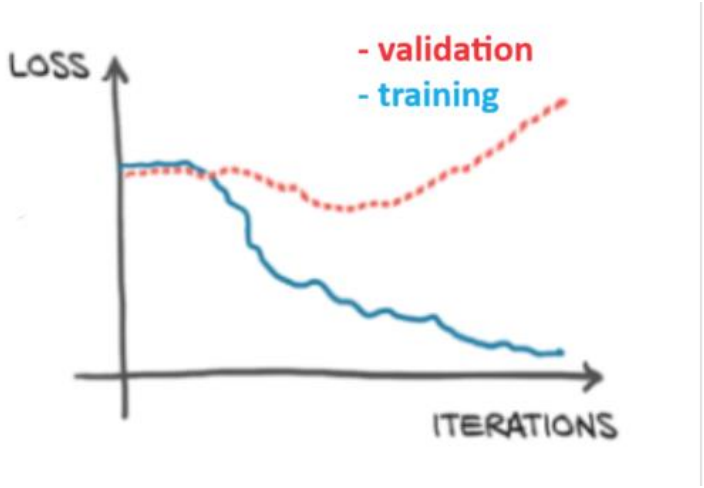
Q4



What is the impact of increasing the batch size in training?

- increases memory usage and training time
- **increases memory usage and decreases training time**
- gives better generalization and guarantees convergence to a global minimum
- makes the gradient estimate noisy, increases the number of weight updates

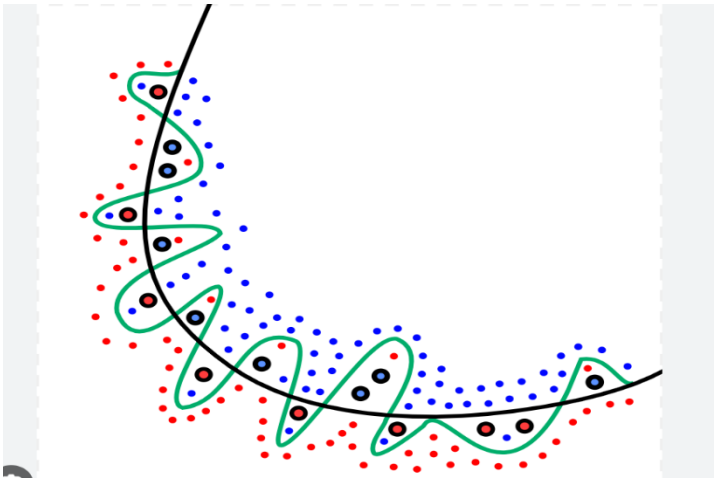
Q5



After training our neural NN, we plot the validation and training loss. What can we conclude?

- Perfect, our job is done.
- We should train for a few more epochs.
- The network was too simple, we should use a more complex one
- **We have a case of overfitting.**

Q6



Which of the following is NOT a cause of overfitting in neural networks?

- **A very low learning rate**
- Using a very deep network with many layers
- Training with insufficient amounts of data
- Training a model for too many epochs

Q7

What is the primary benefit of implementing early stopping during the training of a neural network?

- increases the training speed by using more computational resources
- simplifies the model's architecture by reducing the number of layers
- **prevents overfitting by looking at the validation performance and stopping**
- finds optimal hyperparameters during training

Q8

object in scope. Tip: use ray.put() to put large objects in the ray object store.

Trial Progress

Trial name	should_checkpoint	training_accuracy	training_loss	val_accuracy	val_loss
train_fashion_5f72b_00000	True	0.759042	1.70152	0.783417	1.67756
train_fashion_5f72b_00001	True	0.502333	1.95866	0.5295	1.93167
train_fashion_5f72b_00002		0.749271	1.71175	0.705417	1.75491
train_fashion_5f72b_00003	True	0.833104	1.6461	0.838167	1.63788
train_fashion_5f72b_00004	True	0.835375	1.64392	0.834	1.64186
train_fashion_5f72b_00005	True	0.841375	1.62371	0.835833	1.62754
train_fashion_5f72b_00006	True	0.840729	1.62629	0.849167	1.61692
train_fashion_5f72b_00007	True	0.601104	1.8597	0.611917	1.84866

When using Ray Tune, what is a trial?

- A single mini batch of model training
- **A single set of hyperparameters**
- A single epoch of model training
- A single hold out fold of the training data

Q9

```
ckpt_path='mnist-model-tb/lightning_logs/version_0/checkpoints/epoch=9-step=7500.ckpt'
ckpt = torch.load(ckpt_path)
print(ckpt.keys())
```

```
dict_keys(['epoch', 'global_step', 'pytorch-lightning_version', 'state_dict', 'loops', 'callbacks', 'optimizer_states', 'lr_schedulers'])
```

Which of the following is NOT typically saved in a checkpoint?

- Model's weight parameters
- Model's training hyperparameters
- **Data used during the training process**
- State of the optimizer

Q10

How overwhelming was the video + notebook of neural networks?

- **slightly, but, hey, that's deep learning**
- overwhelming, but not more than usual
- video was ok, code seems hard
- I actually thought we were going to learn more details about NN