Preston Pinh

# Neural Networks

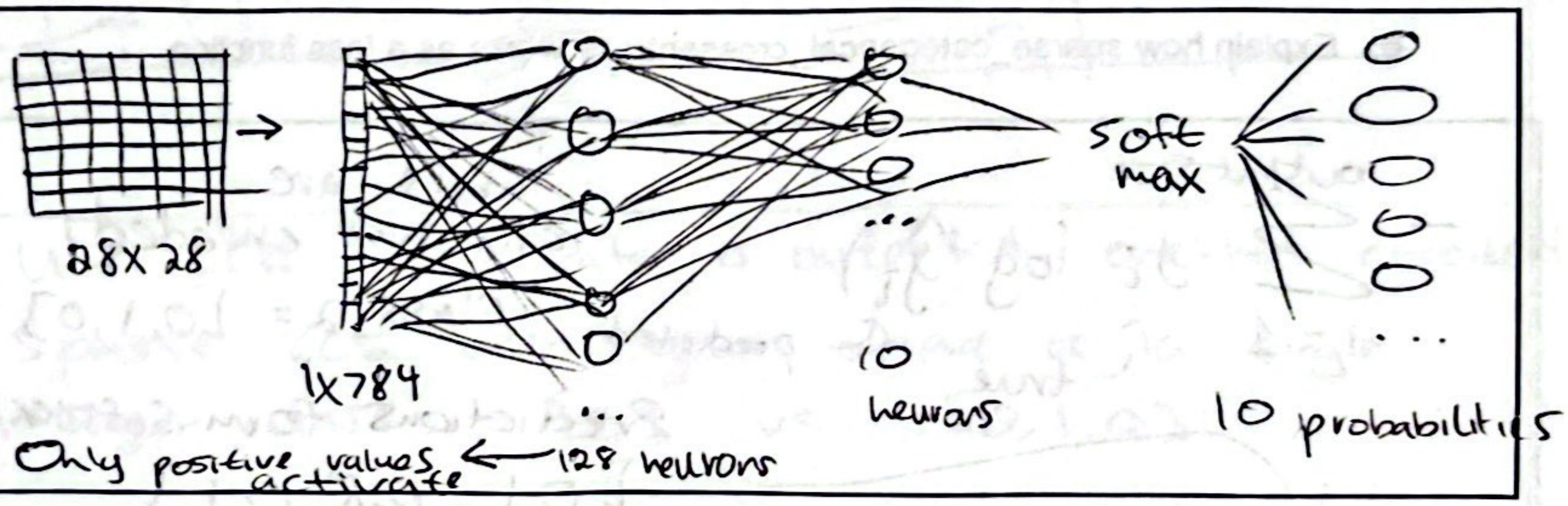
1. Consider the following neural network:

```
model = models.Sequential([
    Input(shape=(28, 28)),
    layers.Flatten(),  # Flatten the 28x28 images into 1D vector
    layers.Dense(128, activation='relu'),  # Dense layer with ReLU activation
    layers.Dense(10, activation='softmax') # Output layer with 10 classes (digits 0-9)
])
```

a. How many layers does this network have?

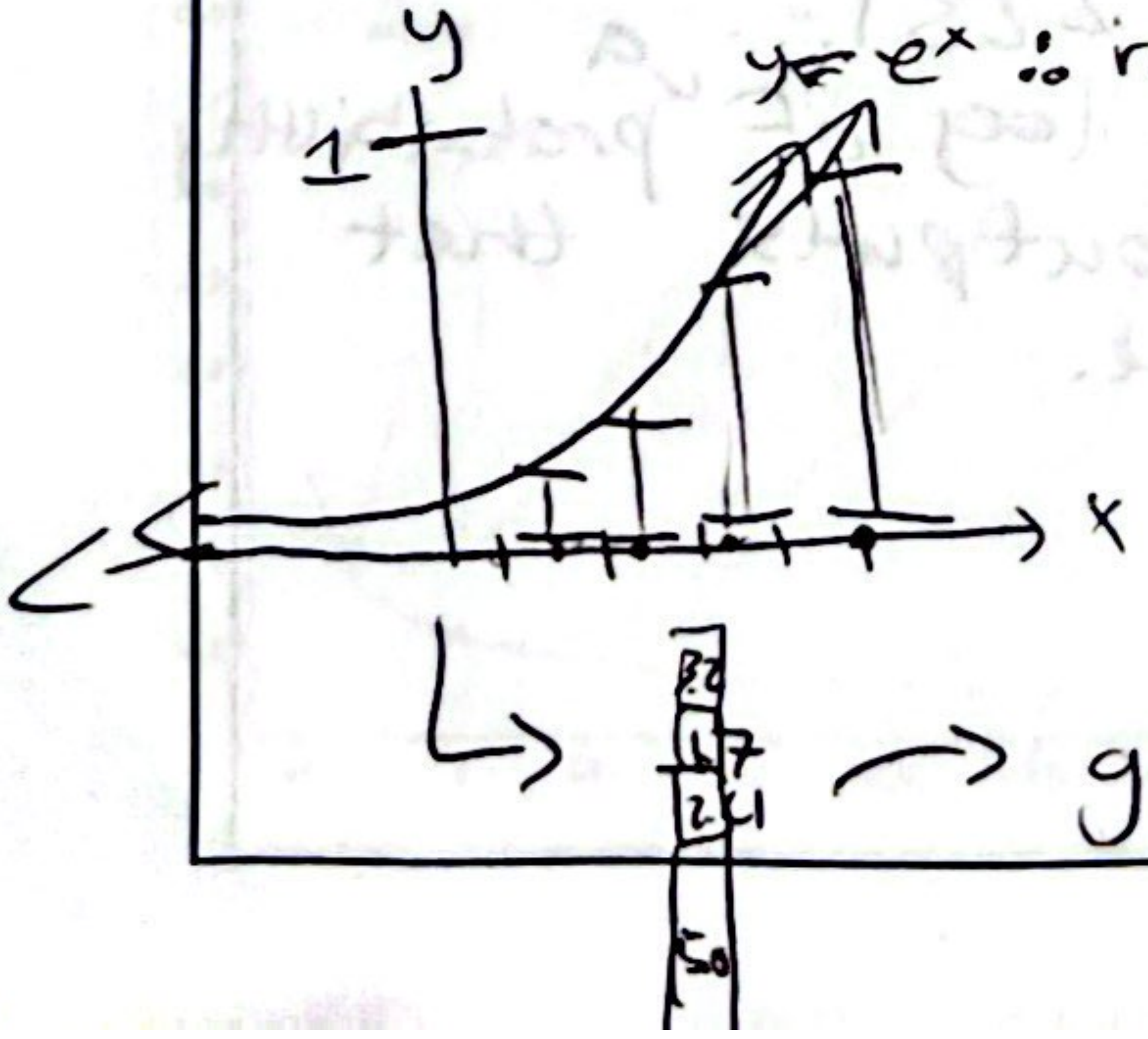
2 Layers

b. Show how the handwritten digit image is transformed as it passes through the neural network layers to produce the required output.



c. Explain how the activation function softmax works to produce the required predicted probabilities for each class. Include the formula for softmax.

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}} \quad \begin{bmatrix} 5.0 \\ 2.4 \\ 1.7 \\ 3.2 \end{bmatrix} =$$



Each x value corresponds to a y-value on the graph of $e^x$ ($e^{z_i}$). That y-value is put over the summation of all y-values, giving a percentage out of 1.

→ gives probability value for each $z_i$ value.

d. **Count the trainable parameters in the network. Show your calculations for each layer.**

129 weights + bias in layer 1

11 weights + bias in layer 2

140 total trainable params

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

e. **Explain how sparse_categorical_crossentropy works as a loss function.**

$$-\sum_{i=1}^{\text{output size}} y_i \log(\hat{y}_i)$$

↑ true ↑ predicted

labels are one-hot encoded:
Class 2 = [0,1,0]

Predictions from softmax = [0.1, 0.8, 0.1]

$y_2 = 1, \hat{y}_2 = 0.8$

$\text{Loss} = -(0 \cdot \log(0.8) + 1 \cdot \log(0.8) + 0 \cdot \log(0.1))$

$-(-0.2231\ldots)$

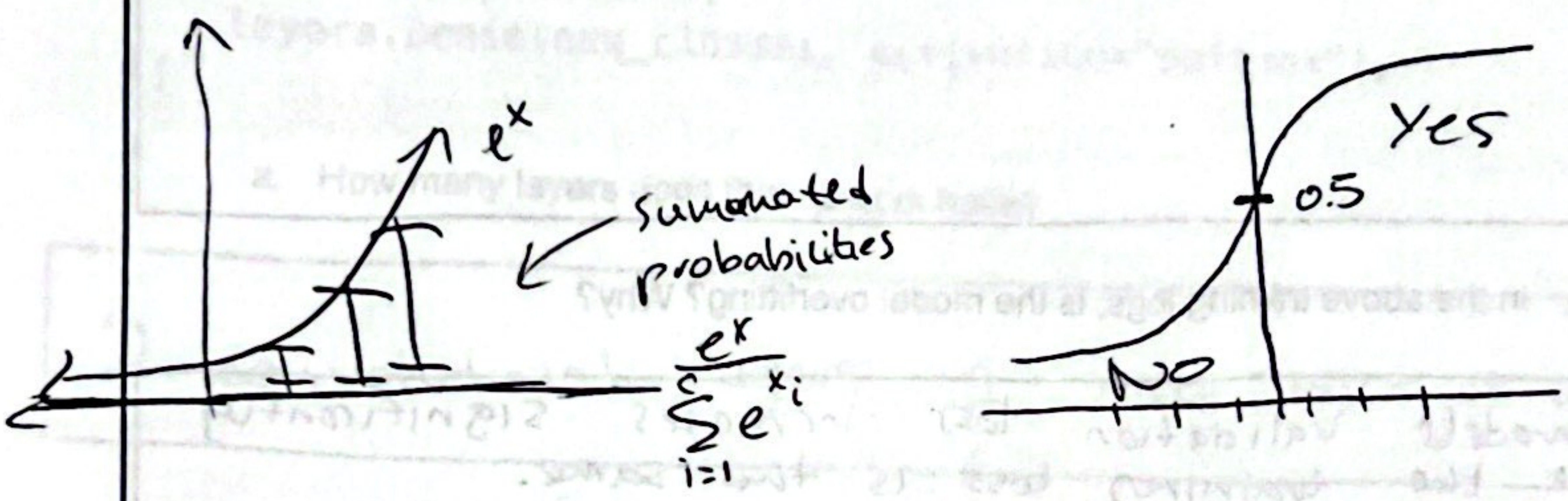
$= 0.2231\ldots$

CCE calculates negative log of a probability of a certain class and outputs that number as a loss value.

f. **Compare binary_cross_entropy with categorical_crossentropy. When is categorical_crossentropy used instead of sparse_categorical crossentropy?**

Binary cross entropy: for when only 2 possible outcomes

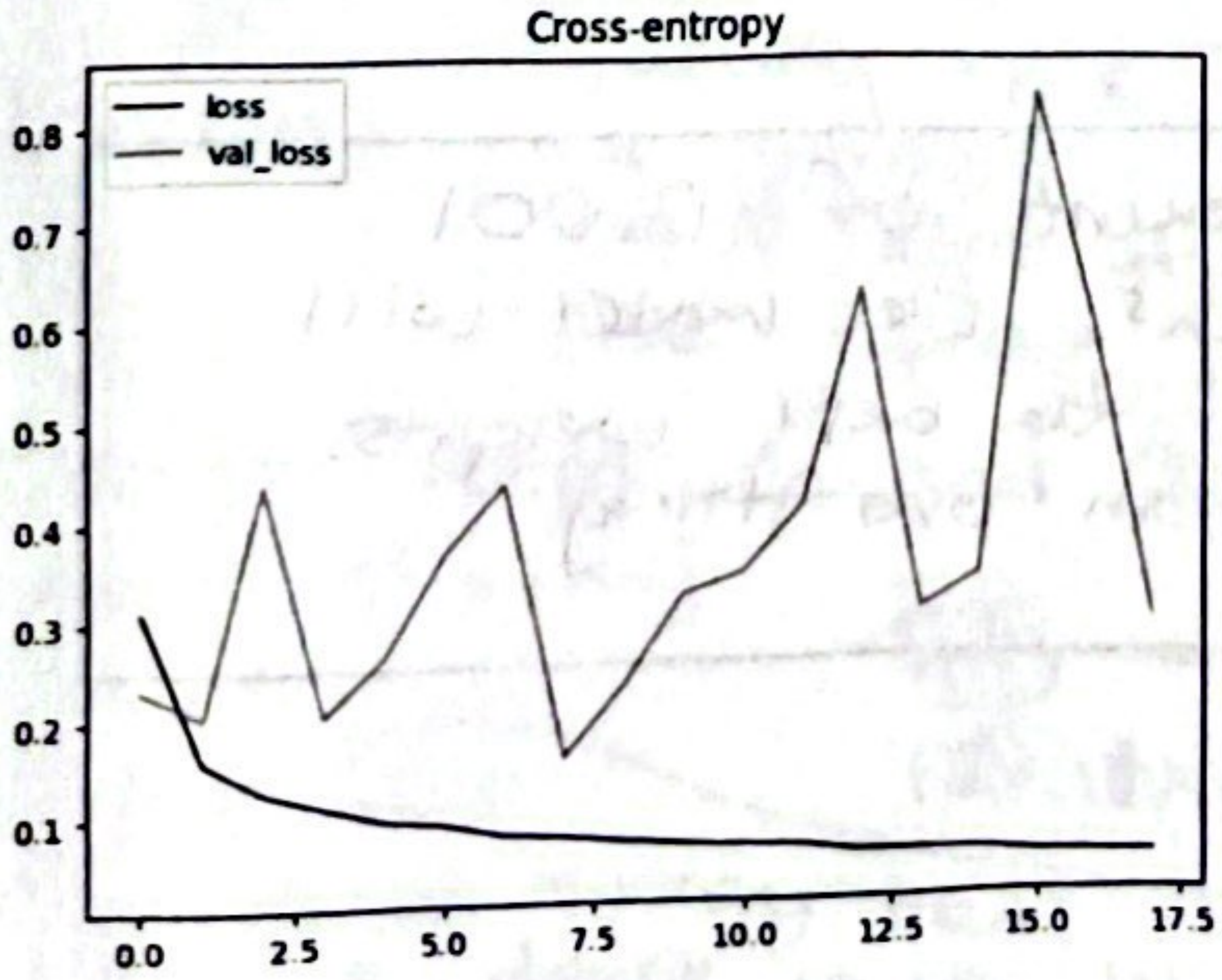
CCE for multiclass classification w/ 3 or more possible categories. Assigns probabilities to each.

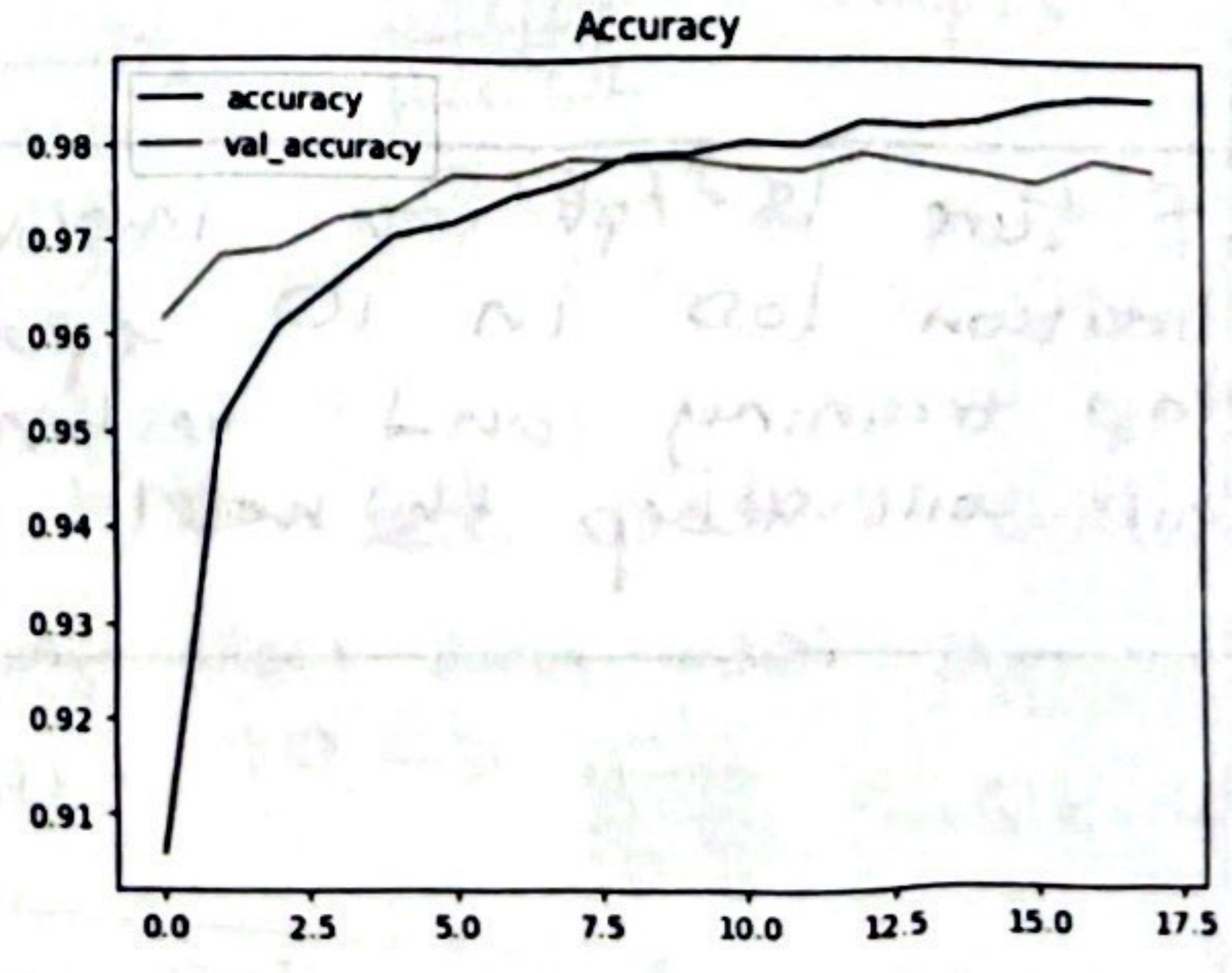


Use CCE when data is outputed one-hot encoded

Sparse CCE when your output is a single integer class. [2] vs. [0,1,0]

g. **What is the difference between the validation loss and validation accuracy? In this diagram below, the validation loss is widely different around epoch 13, but the validation accuracy is about the same. Explain why this happens.**





In calculating loss, -log(probability), it may be high if your probability distribution is relatively similar [0.[illegible], 0.3, 0.2, 0.2, 0.2].
However if the correct class is 2, the model is still correct, so the accuracy is favorable but the loss is high.

h. In the above training logs, is the model overfitting? Why?

The models validation loss increases significantly while the training loss is the same.

i. Explain how early_stopping callback function can be used to stop the model from overfitting in the following code:

```
early_stopping = tf.keras.callbacks.EarlyStopping(
    patience=10,
    min_delta=0.001,
    restore_best_weights=True,
)
```

If there is not an improvement ~~of~~ 0.001 validation loss in 10 epochs, the model will stop training and restore the best weights. This will keep the model from overfitting

## Convolutional Neural Networks

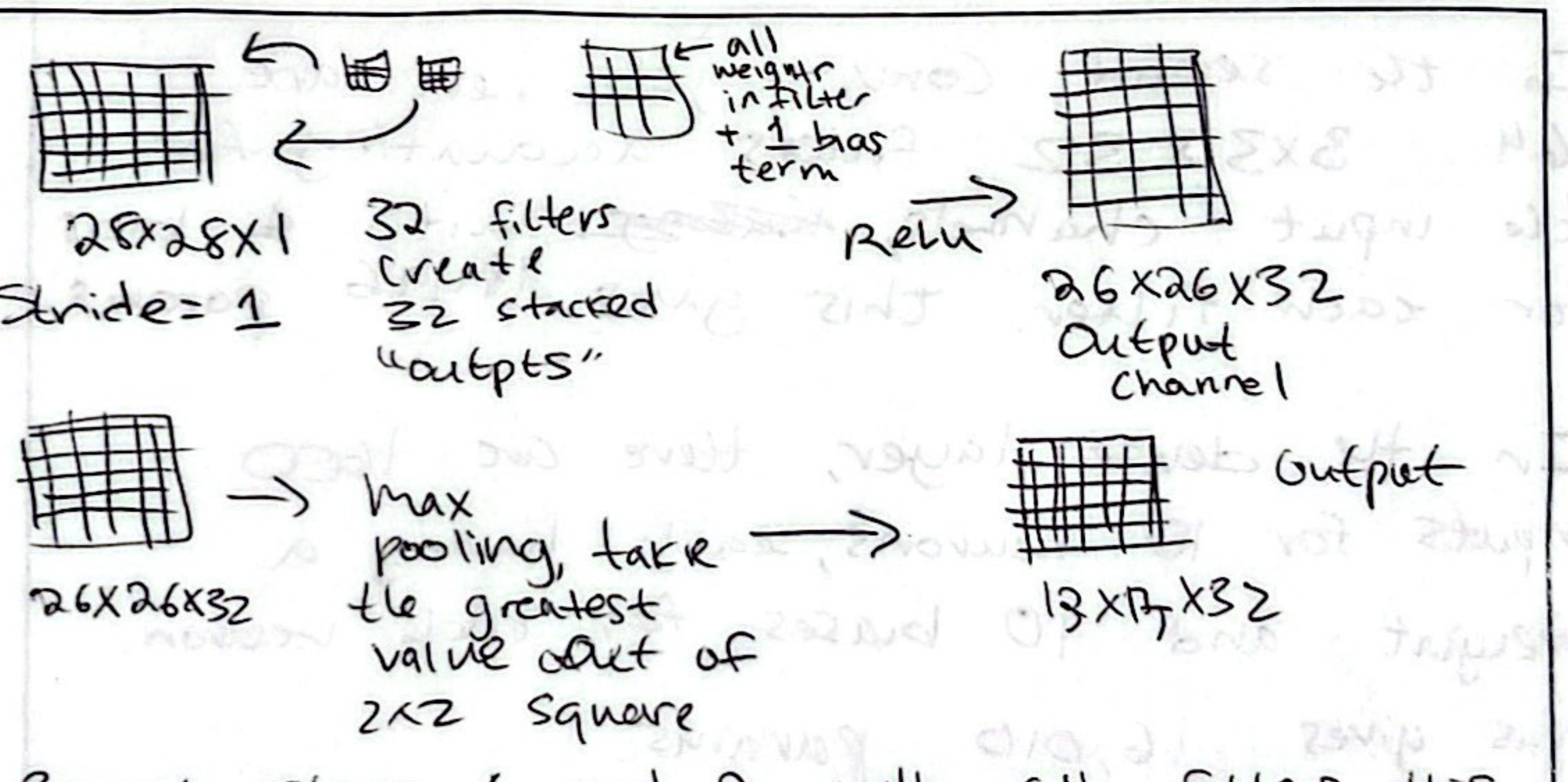
1. Consider the following CNN:

```
model = keras.Sequential([
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ])
```

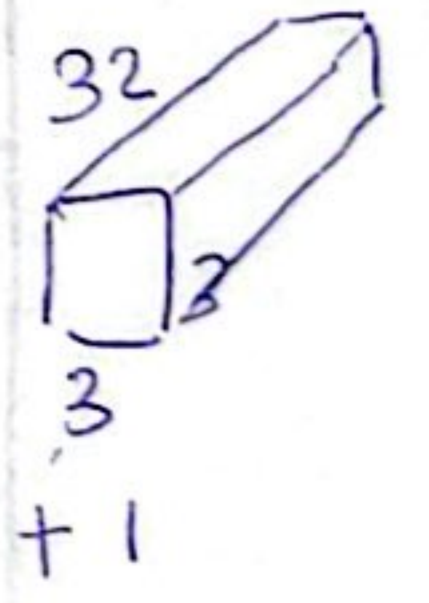
a. How many layers does this network have?

2 Convolutional layers, 1 dense layer of num classes

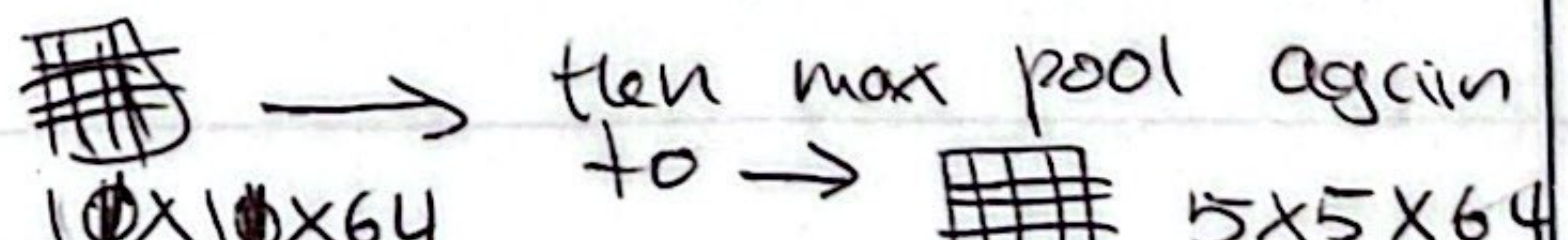
b. Show how the handwritten digit image is transformed as it passes through the neural network layers to produce the required output.



all weights in filter + 1 bias term
28x28x1
Stride = 1
32 filters create 32 stacked "outputs"
Relu
26x26x32
Output channel
26x26x32
max pooling, take the greatest value out of 2x2 square
Output
13x13x32

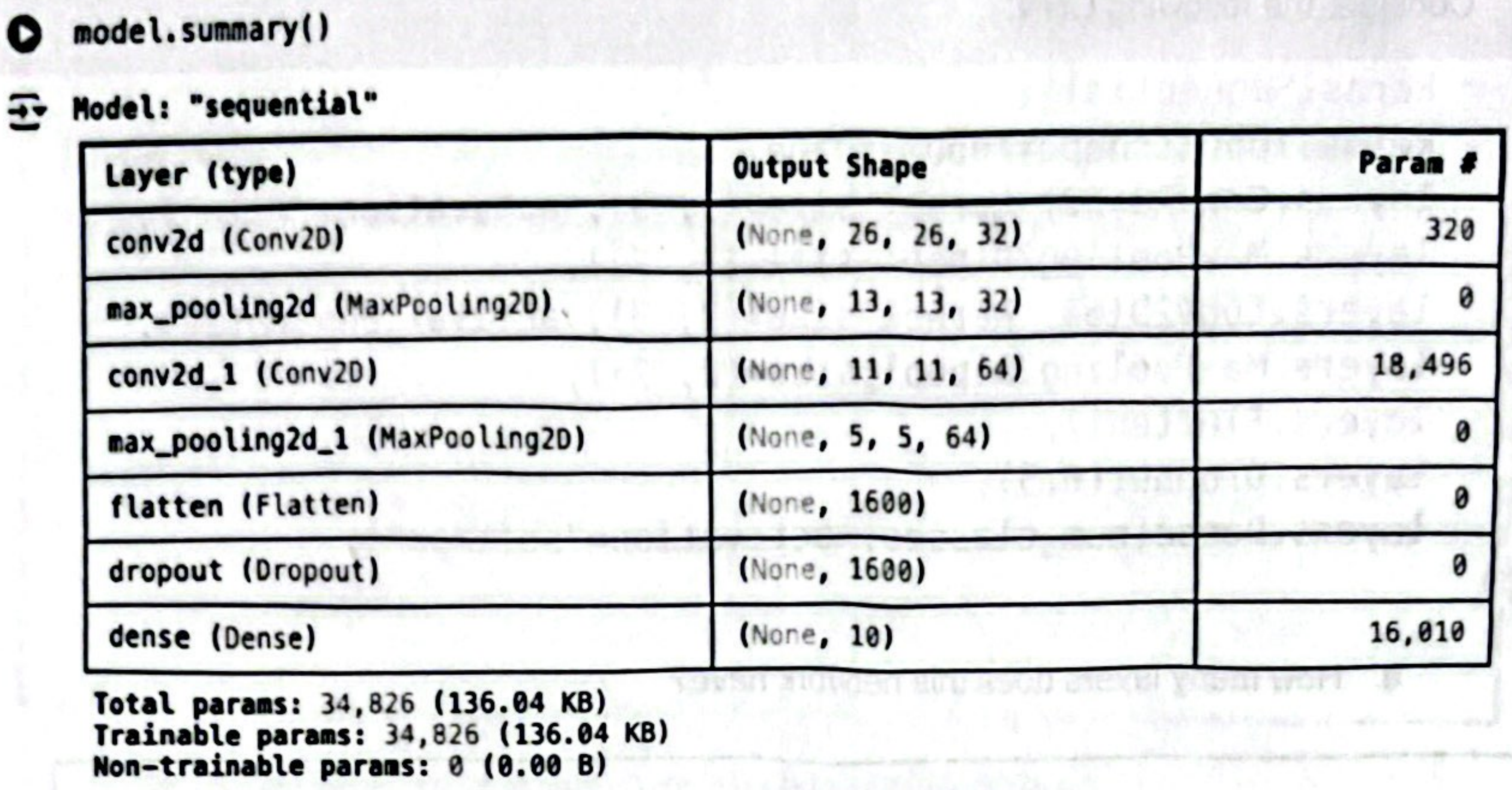


Repeat steps 1 and 2 with 64 filters this time. Giving



Flatten this into a single array then applies a dense layer with softmax for probabilities with a 0.5 dropout rate.

c. **Count the trainable parameters in the network. Show your calculations for each layer.**



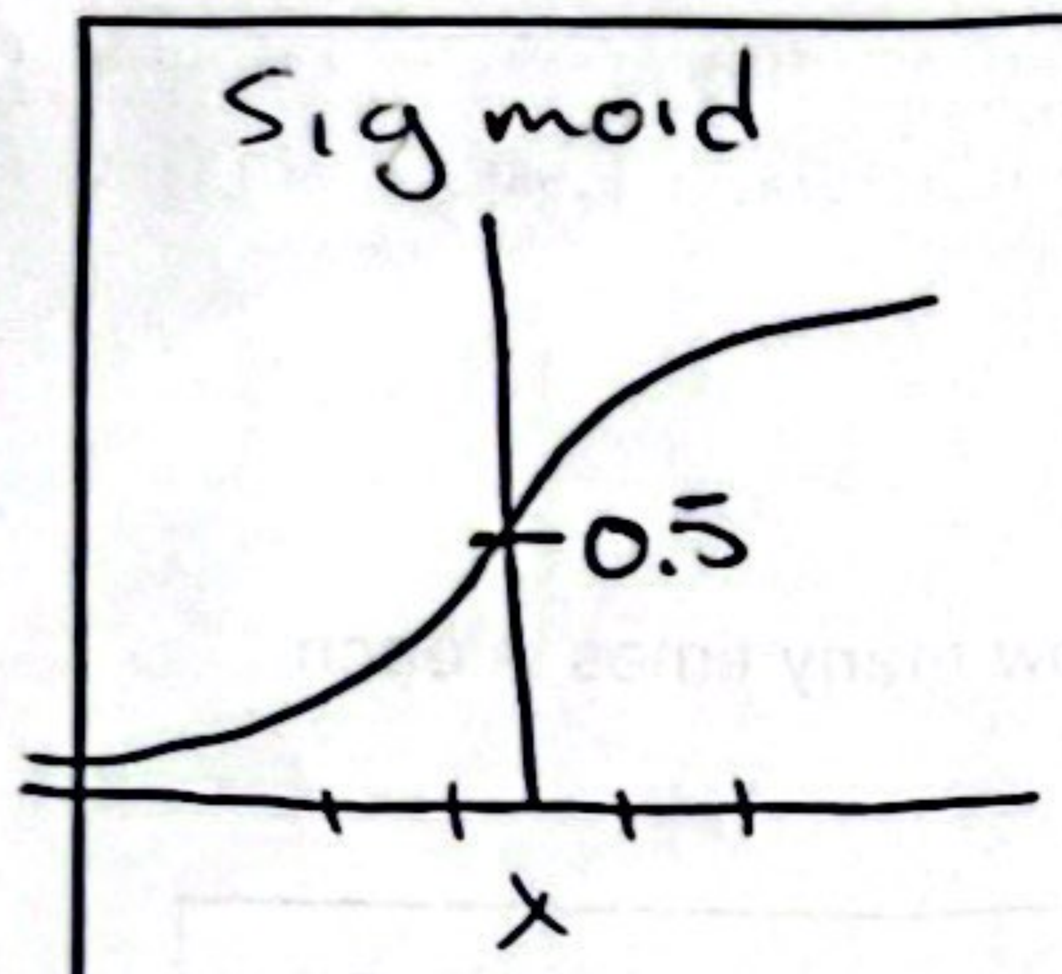
```
model.summary()

Model: "sequential"
┌──────────────────────────────────┬──────────────────────┬───────────┐
│ Layer (type)                     │ Output Shape         │   Param # │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ conv2d (Conv2D)                  │ (None, 26, 26, 32)   │       320 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 13, 13, 32)   │         0 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ conv2d_1 (Conv2D)                │ (None, 11, 11, 64)   │    18,496 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 5, 5, 64)     │         0 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ flatten (Flatten)                │ (None, 1600)         │         0 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ dropout (Dropout)                │ (None, 1600)         │         0 │
├──────────────────────────────────┼──────────────────────┼───────────┤
│ dense (Dense)                    │ (None, 10)           │    16,010 │
└──────────────────────────────────┴──────────────────────┴───────────┘
Total params: 34,826 (136.04 KB)
Trainable params: 34,826 (136.04 KB)
Non-trainable params: 0 (0.00 B)
```

Each fiter has ~~[illegible]~~9 weights and 1 bias, for 32 filters, thats 320 parameters.

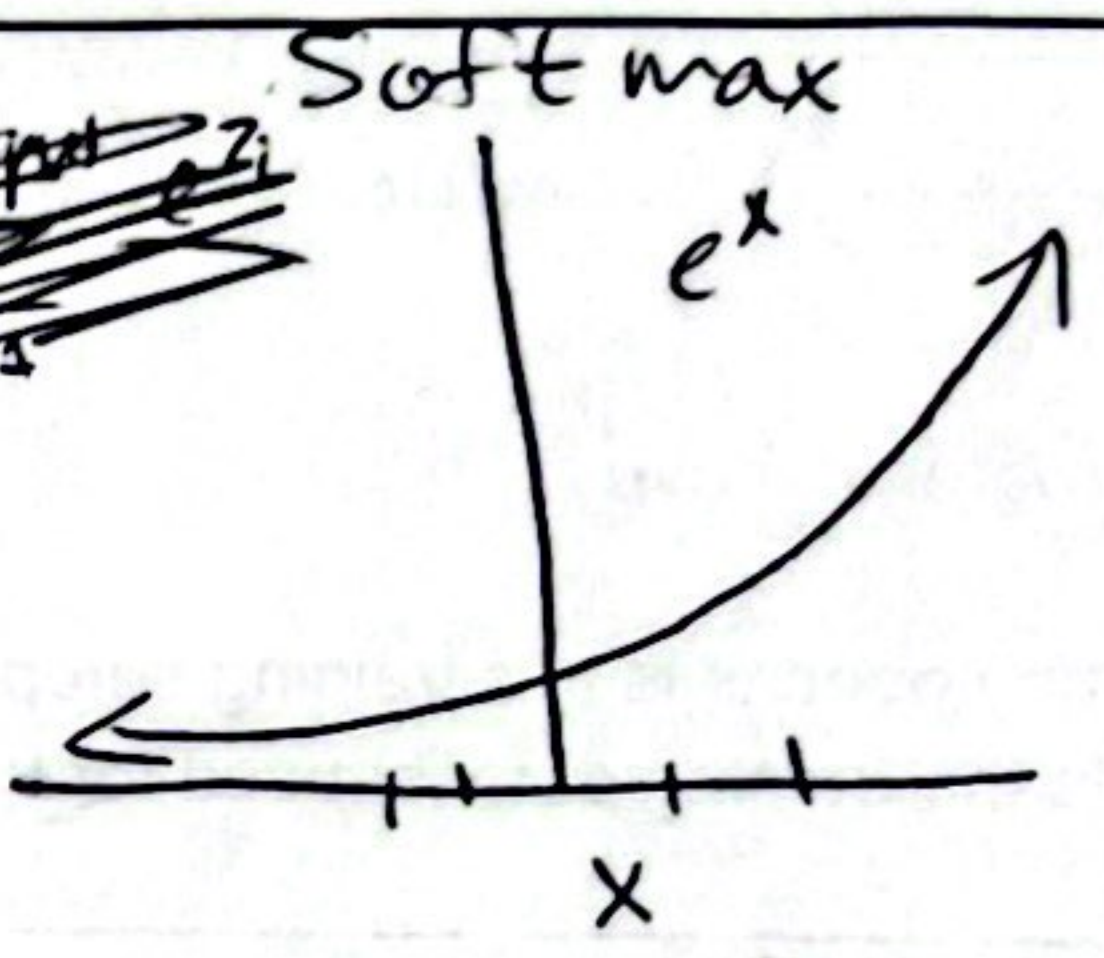
In the second Conv. layer, there are 64 3x3X32 filters accounting for the input channels, ~~[illegible]~~ with 1 bias for each filter. this gives 18496 params.

In the dense layer, there are 1600 inputs for 10 neurons, each having a weight and 10 biases for each neuron. This gives 16,010 params.

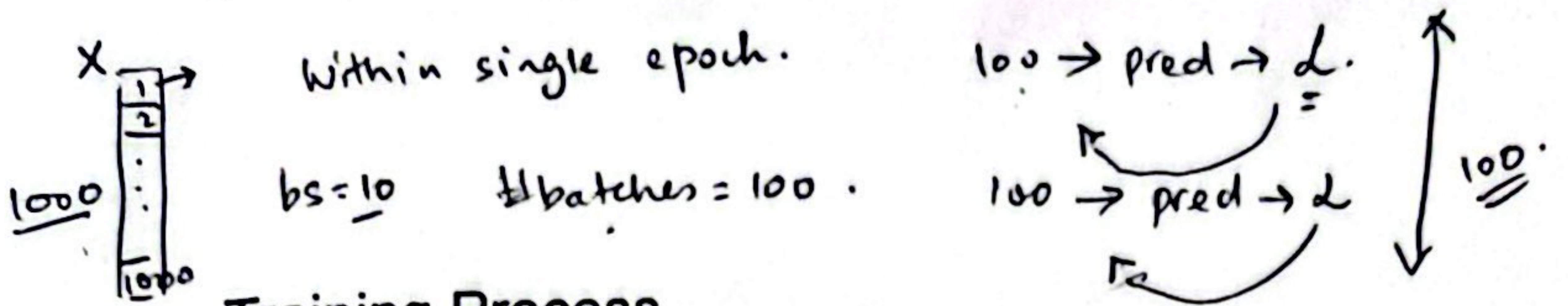
d. **What is the difference between the sigmoid and softmax activation function?**



output $\frac{e^{z_i}}{\sum_{j=1} e^{z_j}}$



Sigmoid will output some value between 0 and 1 that is either greater than or less than 0.5. this is used in binary Classification since that probability will indicate "yes" or "no". Softmax gives a proportion of an ~~x~~ value on $e^x$ over the total x values, with a larger $Z_i$ giving a higher proportion. This gives a probability of 1 $Z_i$ value with respect to other $Z_i$ values.



## Training Process

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 16s 5ms/step - accuracy: 0.8653 - loss: 0.4408 - val_accuracy: 0.9817 - val_loss: 0.0587
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step - accuracy: 0.9723 - loss: 0.0900 - val_accuracy: 0.9863 - val_loss: 0.0413
Epoch 3/20
```

Consider the training progress above:

1. **How many epochs is this training setup to do? In each epoch how many times is each image in the training dataset used for training?**

20 - epochs each image is seen once. Entire training data is seen once.

2. Given the batch size is 32, explain what the number 1875 stands for in the above training progress.

1875 is the amount of batches the model goes through to complete the training set. Training set = 60,000.

3. How many times are the weights and biases updated by the optimizer in each epoch? Which hyper parameter defines this?

All weights + bias is updated once every batch. So they should be optimized 1875 times.