FCNNs

- The number of trainable parameters in a FCNN layer (accounting for bias) is
  - **Total Parameters=(Number of Input Neurons+1)×Number of Output Neurons**
- Without bias, just don't add the +1
- Classification models with K classes has a final layer with K outputs
  - With binary regression, you can output just one layer and use sigmoid activation to get a probability of being one class or the other
- Regression models have just one output
- Model training steps are define, compile, fit, predict, evaluate

Regularization

- Making your model simpler (reducing the number of trainable parameters, either by removing them or penalizing greater model complexity) to address overfitting to the training data so that your model generalizes better to unseen data
- Methods include reducing the number of layers, reducing the number of parameters in layers, early stopping, L1/L2 regularization, and dropout regularizatio

CNNs

- The number of trainable parameters in a CNN layer (accounting for bias) is
  - **Total Parameters=(Kernel Width×Kernel Height×Number of Input Channels+1)×Number of Output Channels**
- In a CNN, assuming that the input shape is **n_h×n_w** and the convolution kernel shape is **k_h×k_w** , then the output shape will be
  - **(n_h − k_h + 1)×(n_w − k_w + 1)**
- In general, if we add a total of **p_h** rows of padding (roughly half on top and half on bottom) and a total of **p_w** columns of padding (roughly half on the left and half on the right), the output shape will be
  - **(n_h − k_h + p_h + 1)×(n_w − k_w + p_w + 1)**.
- In general, when the stride for the height is **s_h** and the stride for the width is **s_w**, the output shape is
  - ⌊**(n_h − k_h + p_h + s_h) / s_h**⌋×⌊**(n_w − k_w + p_w + s_w) / s_w**⌋.
- Pooling, dropout, and flatten layers have no trainable parameters
- LeNet is an architecture that utilizes two convolutional layers, two pooling layers, and three dense layers
- Receptive field refers to all the elements (from all the previous layers) that may affect the calculation of x during the forward propagation

Transformers

https://docs.google.com/presentation/d/1T3evISjLL9XYBqOIWf5Jy_jhslvvYI9E/edit#slide=id.p13

- Input and output are the same size
- Number of parameters does not increase with the input length
- Compute self attention and then add positional encoding
- Can be multi-headed (input is split up, and outputs are concatenated)

- Transformers have residual connections (input is processed, then the output is x’ord with the original input)
- Tokenizers choose the input unit