Describe in your own words (one-two paragraphs, half a page max) what problems an LSTM architecture addresses and how. [3 points]

LSTMs address long-term dependency problem in RNNs. This essentially helps in remembering (filtering out) important information over long sequences. In a simple RNN, as the length of the sequence increases, there are a greater number of terms in the gradient and the further away a term is from the current timestep, the lesser contribution it has in the gradient calculation. This leads to loss of essential information. To address that, LSTMs use the concept of gates.

A single LSTM cell consists of 3 gates: Forget (decides what part to erase), Input (decides potential candidates for long-term memory and what parts of it to have) and Output gates (decides the new hidden state). It's a complex architecture of these three gates which uses pointwise mathematics to compute the results. Using this architecture, LSTMs address two major issues:

1. **Vanishing Gradient Problem**: This happens when ||W||<1; during backpropagation the weights become very small and hence the product of multiple derivatives (chain rule) tends to a very small value. The RNN forgets words at beginning as the sentence gets longer.
2. **Exploding Gradient Problem:** This happens when ||W||>1; during backpropagation, the product explodes (becomes very large value).

LSTMs inherently normalize the output and input values (multiple sigmoid activation functions at each stage) and use the concept of Long-Term Memory (using gates) and point wise product, leading to better retention over long sequences.

Yin et al. paper describes experiments involving various neural architectures. Describe in your own words (one-two paragraphs, half a page max) when and why/how GRU outperformed CNN. [3 points]

GRUs outperform CNNs when there is a contextual and/or semantic dependency on the entire sentence. In other words, GRUs are suited for tasks where word order matters, sentence structures are complicated (like double negatives) and global context overweighs local context.

These tasks include:

1. **Sentiment Classification:** Many sentences could have double negatives like 'I can't imagine hating that.' This sentence is actually a positive sentence, but CNN take key phrases like 'can't' and 'hate' and classify as negative. GRU takes the entire context from left to right and realizes that this is a double negative, so classifies the sentence as positive.
2. **Textual Entailment:** This involves comparing two sentences and telling if one logically follows another. Such a task has a memory dependency on previous sequence and word order, and hence GRU outperform CNN. A CNN will capture both the sentences and key phrases at one go, whereas GRU will do so sequentially – learning the order and sequence.
3. **Sentence Length:** As the length of sentences increase, the chances of complex negations, sarcasms etc increase; GRU perform better at capturing such hidden semantic meanings compared to CNN.

In short, in tasks where there is long-term sequence dependency, GRUs perform better because their architecture supports memory retention using Reset and Update gates.

Apply LSTM on IMDB dataset as we did for last assignment w.r.t. RNN. Report your loss after 100 epochs. [4 points]

I have used Tensorflow and Keras for this task. Tensorflow has an inbuilt dataset called IMDB with integer encoded words. 25000 reviews.

Architecture: Embedding layer -> LSTM layer -> Dense layer -> Output (0/1)

Optimizer = Adam, Loss = 'Binary Cross Entropy'

```
model = Sequential()

model.add(Input(shape = (maxlen, )))
model.add(Embedding(input_dim=num_words, output_dim=128))
model.add(LSTM(64))
model.add(Dense(1, activation = 'sigmoid'))

model.summary()
```

Since this is a classification problem, I have used **sigmoid** as dense layer activation function.

I have restricted max review length to be 200 words for each sentence, and selected only top 10000 highest freq words.

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 1.0000 - loss: 7.1701e-05 - val_accuracy: 0.8622 - val_loss: 1.0642
Epoch 95/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 1.0000 - loss: 5.4374e-05 - val_accuracy: 0.8620 - val_loss: 1.0900
Epoch 96/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - accuracy: 1.0000 - loss: 4.6011e-05 - val_accuracy: 0.8618 - val_loss: 1.1130
Epoch 97/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 1.0000 - loss: 3.4859e-05 - val_accuracy: 0.8618 - val_loss: 1.1379
Epoch 98/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 1.0000 - loss: 2.9919e-05 - val_accuracy: 0.8614 - val_loss: 1.1604
Epoch 99/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - accuracy: 1.0000 - loss: 2.4643e-05 - val_accuracy: 0.8614 - val_loss: 1.1803
Epoch 100/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 1.0000 - loss: 1.8633e-05 - val_accuracy: 0.8616 - val_loss: 1.2024
```

The model started converging from 7th epoch itself toward a near perfect accuracy. But the validation accuracy was around 85%

When the model was tested on unseen test data (for classification task), following results were observed:

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8500 - loss: 1.2685
Loss:  1.2737473249435425
Accuracy:  0.8516799807548523
```

Conclusion: While the accuracy of classification has significantly improved, the model has also started overfitting.