1. How do you observe underfitting of your model and what can you do to prevent this from happening?

Underfitting: error value on training set too large – model does not learn enough from data

Prevention: try a new model / algorithm, fix errors, increase model capacity

2. How do you observe overfitting of your model and what can you do to prevent this from happening?

Overfitting: gap between training error and test error is large – model is learning too much from data and not adapting well to new unseen data

Prevention: regularisation techniques, less parameters, more data, use validation set

3. What is the difference between training, validation and test set and what can low or high error on these sets tell you?

Set		High error	Low error
Training	Used to train model parameters	Underfitting	Model works well
Validation	Used to tune hyperparameters and/or see how model reacts to new data	Overfitting or hyperparameters need to be adjusted	Model works well (both parameters & hyperparameters)
Test	Used to test final model, should not be used for learning	Overfitting	Model works well

4. What are L1 and L2 regularization and how do they affect the network?

L2 – weight decay – sum of squared parameters: penalises heavy weights, distributes weights more evenly

L1 – sum of absolute values of parameters – results in sparsity (and meaningful subsets of features)

5. What are possible ways of augmenting your data?

Images: shifting pixels in each direction, rotating, scaling, flipping, cropping, adding noise

Speech recognition: adjust pitch, speed, pauses

6. How can you use noise to regularize your model?

Can be added at any level:

Input – random noise, improves robustness of NN

Hidden layers – adding noise to weights, reflects uncertainty (assumption of Bayesian inference)

Output – e.g. label smoothing: replace 0 and 1 with ‘soft’ targets to reflect mistakes in gold labels

7. How does dropout relate to model averaging?

Both involve using sub-models to learn

Dropout synthetically creates a set of sub-networks – all different sub-networks which can be formed by removing non-output units. All models share parameters

Model averaging takes the mean of various models

Dropout is an approximation of model averaging

8. Which typical architectures use parameter sharing and how does this improve generalization?

CNNs

Parameter dependencies can be used to express prior knowledge

Fewer parameters – less prone to overfitting

1. Is it reasonable to evaluate a (rare) disease predictor by its accuracy? For the example below: Discuss which evaluation metric would be most suitable and why!

- Recall
- Correct predictions of a particular class

- % true events that were detected = recall

		prediction		
		disease	no disease	
Truth (data)	disease	5	0	5
	no disease	45	50	95
		50	50	100

$$\text{recall} = \frac{5}{5} = 100\%$$

- % true events that were detected = recall

		prediction		
		disease	no disease	
Truth (data)	disease	0	5	5
	no disease	0	95	95
		0	100	100

$$\text{recall} = \frac{0}{5} = 0\%$$

2. A researcher trains a deep neural network. The model performs great on the train set, but rather poorly on the test set. To improve generalization the researcher: adds dropout, adds L1 and L2 regularization, tries different optimizers and learning rates. Finally there is a configuration where the test error gets close to the train error. Now the model is ready to be published! Do you agree? Discuss whether the researcher followed the good practice. What could he/she have done better?

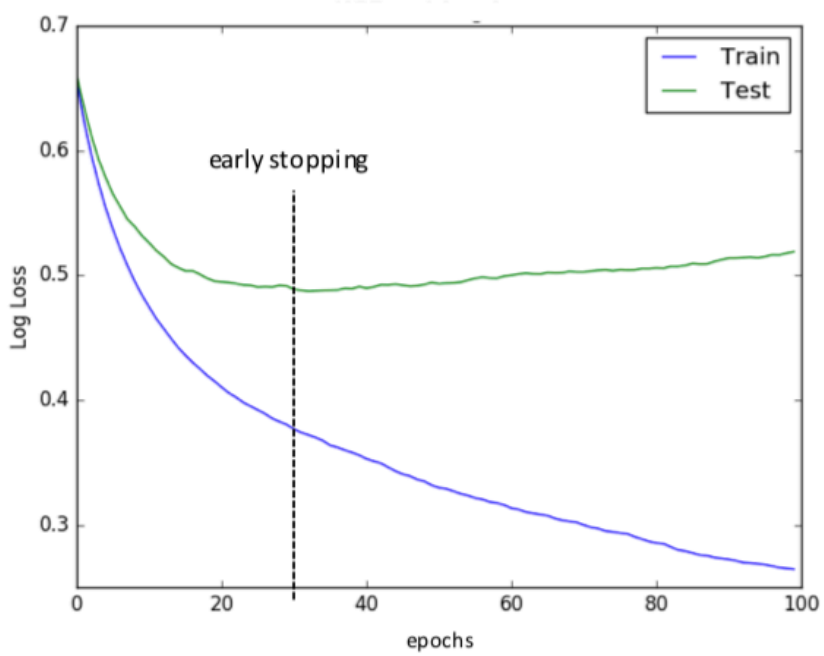
- No, it's not best practice to just look at how the test error relates to the training error and tweak the model until the test error more closely resembles the training error

- The idea of tuning hyperparameters like regularization and optimization, adding dropout, etc. is correct, but this is best done with a validation set

- Train the model on the training set, see how it performs on the validation set, and then perform tuning (like the examples above) based on the validation set results

- Don't directly use test set results to inform your model decisions

3. In some model, we observe that the test loss increases after 30 epochs, therefore we use early stopping at 30 epochs as regularization technique. Discuss whether this is a good approach! If not, think about how to fix the problem!



- Similar to the previous question— don't directly change your model based on test set results

- Can introduce a validation set to find a model tuning that will generalize better on the test set

- Choosing early stopping after 30 epochs here doesn't ensure that this model will generalize well to other unseen data, it's very specific to these test set results