intit param must break symmetry bet. diff. units **init points** determine: 1) whether/how quickly learning coverages; whether the point that it converges to has high or low generalization error. **Adam optimizer** takes velocity into account when taking grad. desc. step. **AdaGrad:** +) not very sens. to init learning rate; +) large partial derv., large learning rate decrease, and vise versa; -)aggressive, monotonically decreasing learning rate **RMSProp:** +) discard history from extreme past; +) less aggressive that AdaGrad; -) has extra p hyperparam. **ADAM:** +) has momentum; +) less biased that RMSProp; -) has 2 hyperparam. p1, p2 **BatchNorm** eases optimiz. by reducing dependence of param bet. layers **BatchNorm:** apply it after non-linearity; layers preceded by it doesn't need bias or dropout. **Receptive field** of an output unit in CNN covers more input if convolution window is wider. **Pooling layer** cannot have their weights updated during training. **CNN** =¿ start [convolution layer I*K] =¿ Dense layer (RELU) =¿ Pooling max(x_a) Convolution: weights of unit i = weights of i+1 but shifted; weights = 0 except for contiguous region; only local interactions, equivariant to translation **Pooling:** invariant to small translation; strong prior leads to potential underfiting RNN's solution to the problem of variable-length sequence is to define the current hidden state as a function of the previous hidden state and the current output. RNN: $h_t = f(h_t - 1, x; W)$ RNN handles variable length sequences; share the same param at each step; incorp. all preceding inputs and hidden states into each decision **hidden2hidden** recurrent connection must exist to compute any Turning-machine computable function. one output per input, and hidden2hidden connection: detecting seizure in EGG signal **BiRNN** classifies YouTube comments as spam; accurately captures local info before and after the current state; can capture long-range info; expensive to train **Seq2Seq:** summarization

1. **The biggest problem for learning long-term dependencies in RNNs is**: Smaller weights are given to long-term interactions 2. **Gated RNNs are better than leaky units because**: They can choose when to forget information 3. **The first neural network model** you train for a supervised learning task should always include: Early stopping 4. If you **don't have time to tune** only one hyperparameter, tune the: Learning rate 5. The code for your performance metric was written by someone else. Which **debugging strategy** is most likely to reveal the bug in this code if there is one? Visualize the model in action

Recurrent neural networks 1 **output per input** Can compute any function that is computable by a Turing machine, training time and memory are O(T) and not parallelizable since all states are needed for backprop, testing time is O(T) and not parallelizable, testing memory is O(1); past states can be discarded, powerful but expensive **Output-to-hidden**: Cannot simulate a universal Turing machine, training is parallelizable with teacher forcing ie using the ground truth output, since then each time step can be calculated independently, testing time is O(T) and not parallelizable, testing memory is O(1); past states can be discarded, may predict poorly when ground truth objects are no longer available (solution: mix training on ground truth outputs with training on predicted outputs) 1 **output at end**: Training time and memory are O(T) and not parallelizable since all states are needed for backprop, testing time is O(T) and not parallelizable, testing memory is O(1); past states can be discarded, can be hard to train on long sequences since output gradient must propagate back through the whole sequence

Back prop through time: Unroll RNN to produce computational graph including all time steps, perform regular back-prop on the graph

Graphical models: edge=dependence != operation **RNNs** efficiently parameterize the joint probability distribution **RNNs as graphical model challenges**: Through parameterization of joint prob dist is efficient, optimizing those params may be difficult; Assumes relationship between adjacent time steps, does not depend on t; Generating samples from an RNN requires extra machinery; Can add special stop output symbol, can add binary stop/dont stop output to each step, can add integer (T) output to each step

Bidirectional RNNs Problem: recurrent neural networks assume that a state depends only on previous states. **Solution**: run one RNN forward, one RNN backward, and combine the two **Merging options**: merge, sum, concatenate. **Bidirectional recurrent nn**: Accurately capture local info both before and after current state (similar to convolutional nn); Can capture some long-range info, both before and after current state; Are expensive to train (two RNNs)