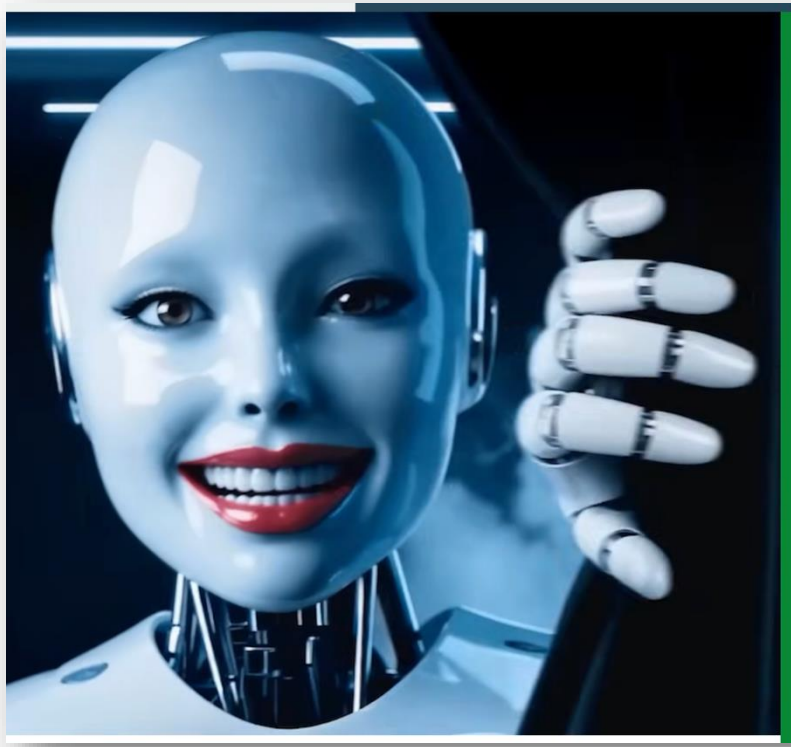
Neural Networks: Foundations to Generative AI

Model Fitting & First Examples

What's In the News?



MEDIA & ENTERTAINMENT

From Svedka to Anthropic, brands make bold plays with AI in Super Bowl ads

Lauren Forristal — 8:18 AM PST · February 8, 2026

This Week's Agenda

News

First Hands-on Examples

- Predicting labels in synthetic data and fitting pure noise.
- Boston Housing home-value prediction.
- Predicting review valence / stars from text.
- Detecting fake reviews.

Dealing with Overfitting

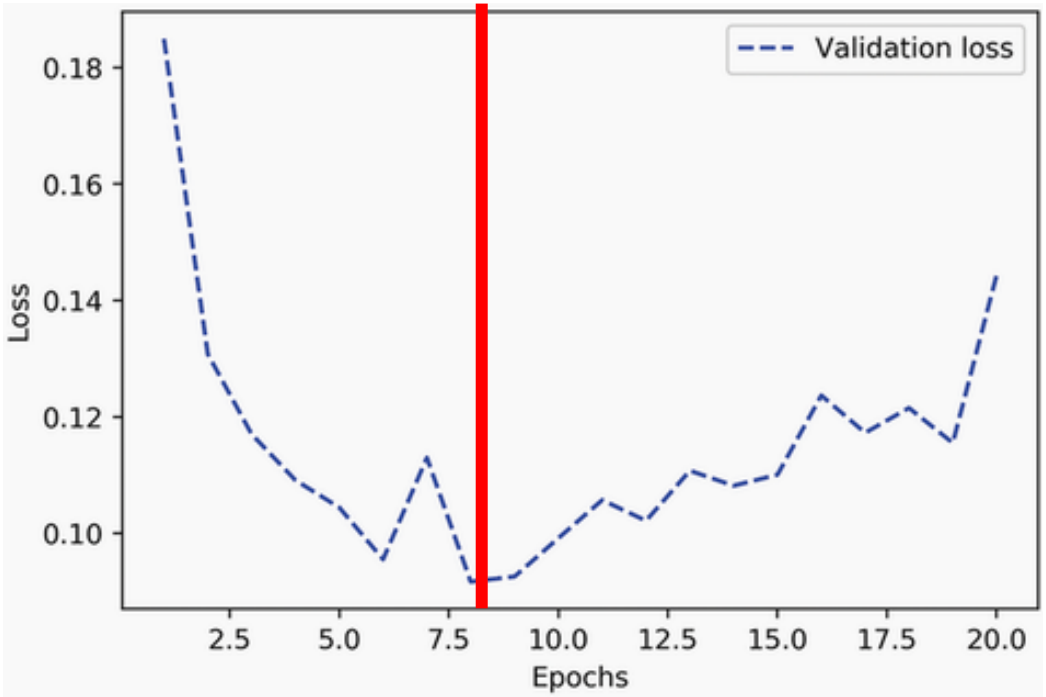
Introducing the Individual Coding Assignment

More on the Final Project Proposal

Early Stopping

Monitoring Validation Performance and then Manually Limiting Epoch Count

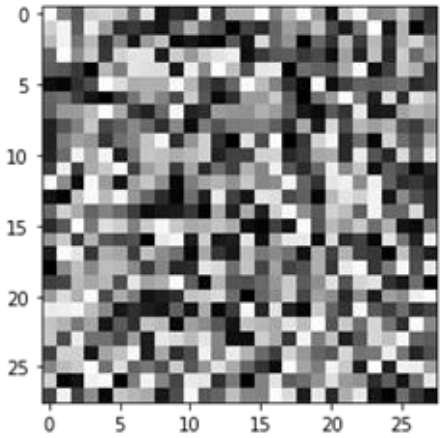
- This is the sole approach we have been taking thus far. Can implement automatically using Keras callbacks (more on this later).

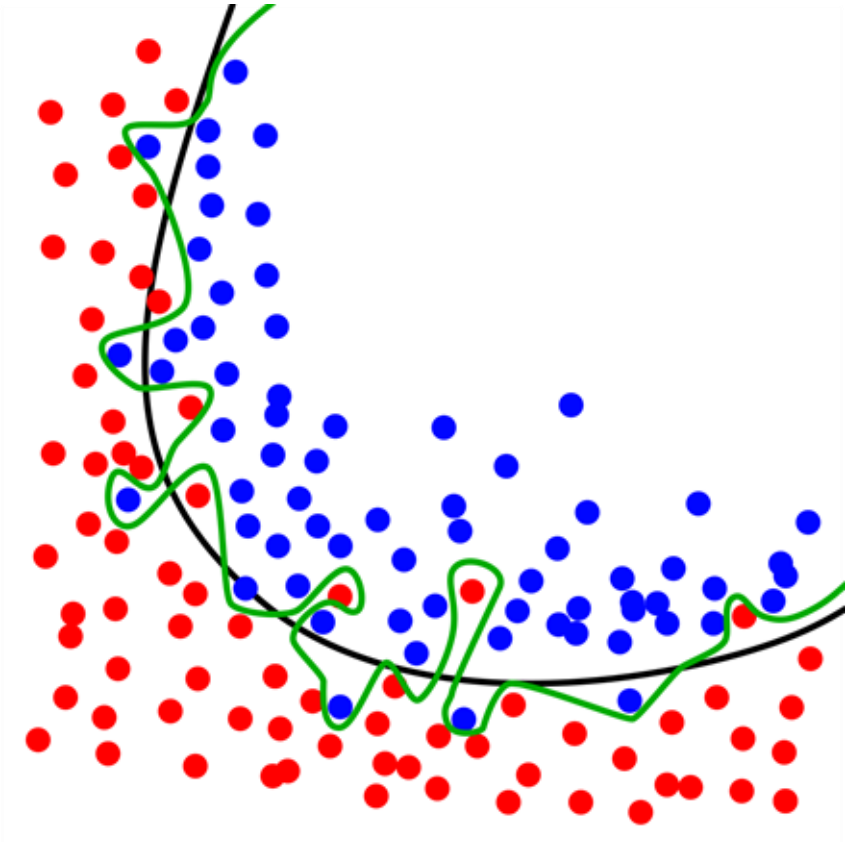


Overfitting

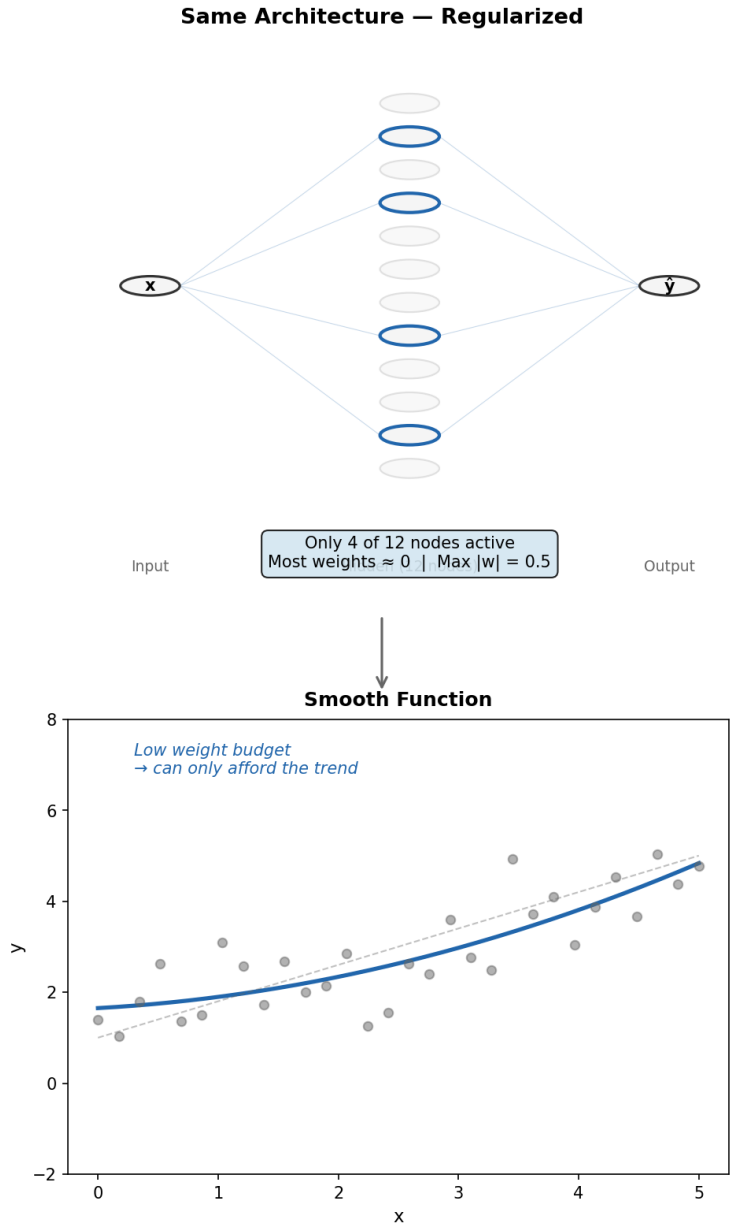
A Neural Network Can Easily Overfit…

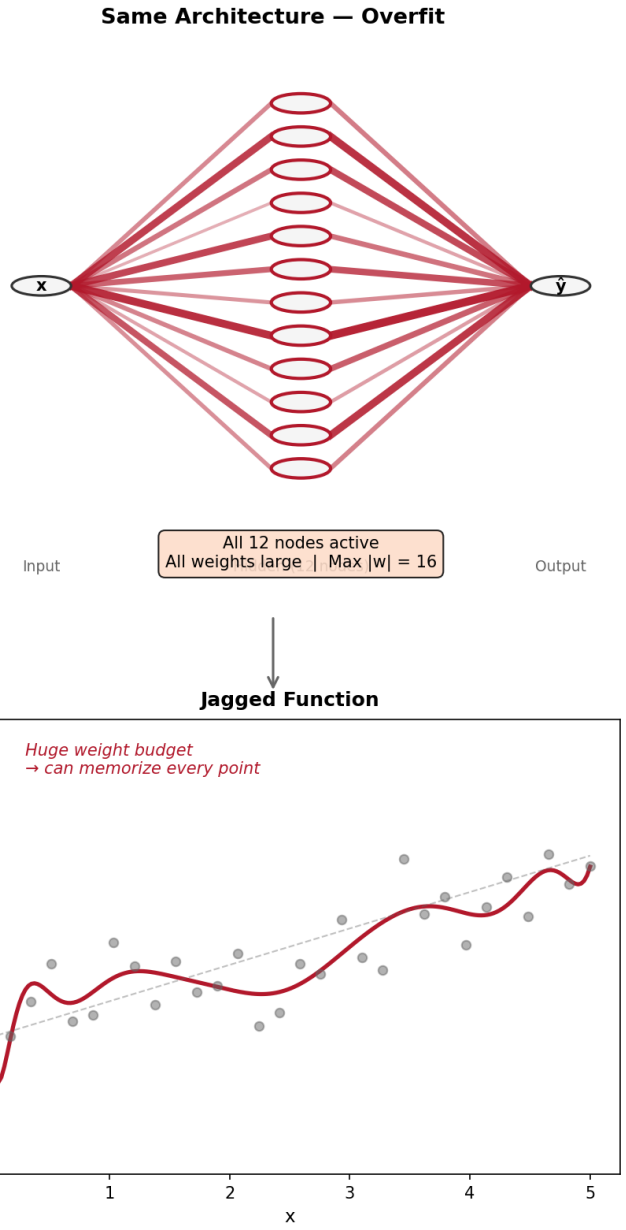
- Let's look at an extreme case…





Overfitting Means Bigger Weights!





Regularization Keeps Weights Small

Regularization via Weight Constraints

- Regularizing a layer's weights means the model is less inclined to increase big weights. Both L1 or L2 norms can be used here applied.

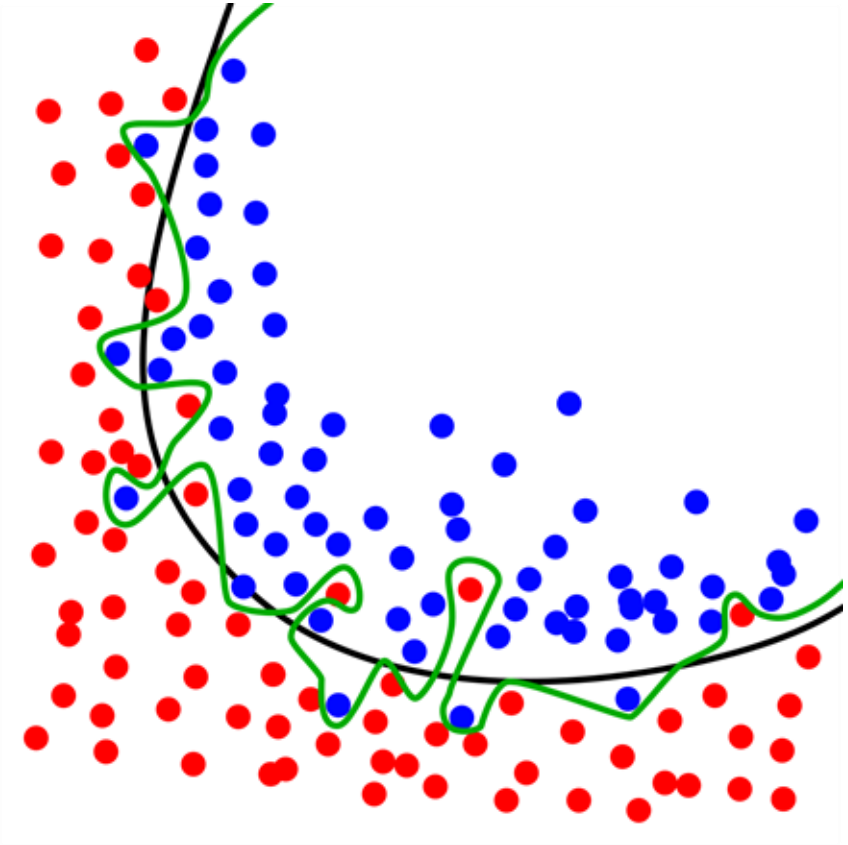
Listing 5.13 Adding L2 weight regularization to the model

```
from tensorflow.keras import regularizers
model = keras.Sequential([
    layers.Dense(16,
                 kernel_regularizer=regularizers.l2(0.002),
                 activation="relu"),
    layers.Dense(16,
                 kernel_regularizer=regularizers.l2(0.002),
                 activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
history_l2_reg = model.fit(
    train_data, train_labels,
    epochs=20, batch_size=512, validation_split=0.4)
```

What ‘Enables’ Overfitting?

Idiosyncratic Noise in Data

- Over-fitting is driven by the network ‘latching on’ to idiosyncrasies in the training data that happen to associate with certain outcomes.
- To solve this, in training, we can add noise / perturbations to the training data, to prevent it from latching onto those idiosyncrasies, so it sticks to more general rules.
- Note: with too much noise, learning general rules gets hard!

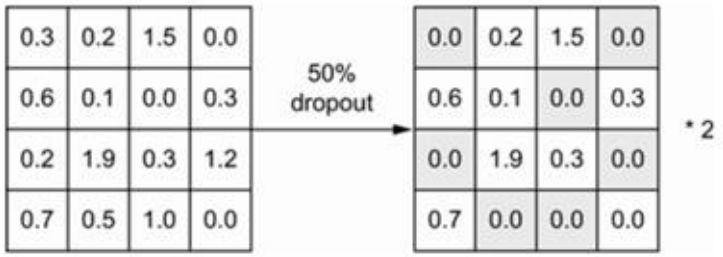


Dropout

Adding Dropout Layers to the Network

- Dropout layers set a random proportion of edge weights to 0 in a given training iteration. Typically, between 20% and 50% of edges.
- Note that this 'zeroing out' of weights is not retained in the final trained model.

Figure 5.20 Dropout applied to an activation matrix at training time, with rescaling happening during training. At test time the activation matrix is unchanged.



Batch Normalization

Re-scales batch coming into each layer, to make it unit-normal distributed.

- The primary reason for these layers is to improve/speed up training. Absent this, changes to up-funnel weights cause ‘shifts’ in the distribution of data entering later layers. Later layers’ weights need to be updated constantly to adapt to those up-stream shifts. Batch normalization removes this issue.

- A by-product of batch normalization is that it acts like a weak form of Dropout, because normalization is implemented based on the current batch of data. Mean and variance estimates from the current batch are a noisy reflection of population mean and variance. Hence, we get some noise being injected in the training process as a result.

Adjust Batch Size

Deep Learning

Ian Goodfellow
Yoshua Bengio
Aaron Courville

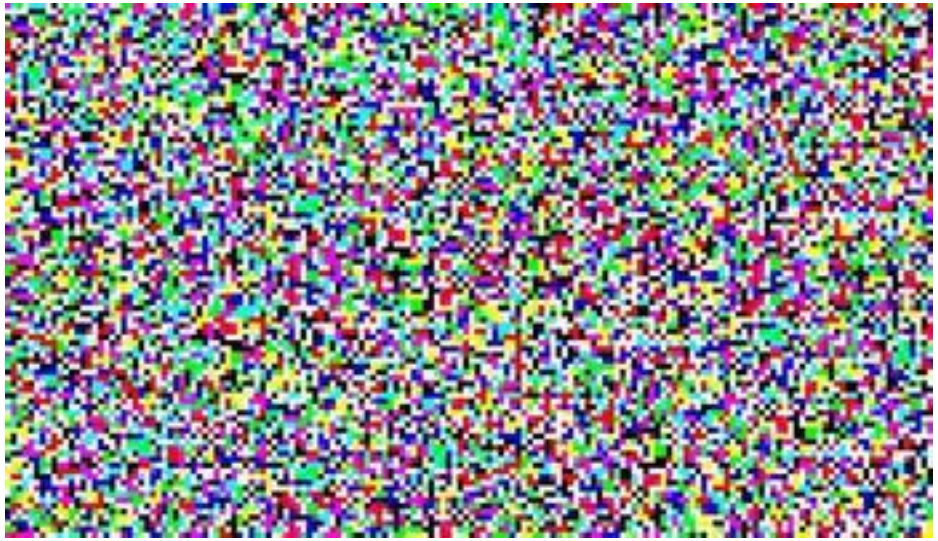
Minibatch sizes are generally driven by the following factors:

- Larger batches provide a more accurate estimate of the gradient, but with less than linear returns.
- Multicore architectures are usually underutilized by extremely small batches. This motivates using some absolute minimum batch size, below which there is no reduction in the time to process a minibatch.
- If all examples in the batch are to be processed in parallel (as is typically the case), then the amount of memory scales with the batch size. For many hardware setups this is the limiting factor in batch size.
- Some kinds of hardware achieve better runtime with specific sizes of arrays. Especially when using GPUs, it is common for power of 2 batch sizes to offer better runtime. Typical power of 2 batch sizes range from 32 to 256, with 16 sometimes being attempted for large models.
- Small batches can offer a regularizing effect (Wilson and Martinez, 2003), perhaps due to the noise they add to the learning process. Generalization error is often best for a batch size of 1. Training with such a small batch size might require a small learning rate to maintain stability because of the high variance in the estimate of the gradient. The total runtime can be very high as a result of the need to make more steps, both because of the reduced learning rate and because it takes more steps to observe the entire training set.

Gaussian Noise

You can use this layer to simply inject noise into the data in the middle of the network.

- There are ways to add noise to any model component, including the inputs, weights and activations.



Get More Data (Idiosyncrasies Play Lesser Role)!

More Training Examples

- Means you can have a bigger training data-set, which will presumably contain more information for the model to extract and any given noise players a weaker role. This is the approach that often yields the best marginal returns, though it can also be most costly.

Some Rules of Thumb

These Are Useful Guidelines for Your First Pass at a NN

- Start with 2 hidden layers.
- Give the first hidden layer (num_inputs / 2) nodes and the next (num_inputs / 4) nodes. If you add layers, decay the node count in this manner as you go along.
- Use ReLU activations for hidden layers.
- Add Dropout at every layer after the input, with a rate of 0.5 (don't push beyond 0.5).
- If learning is flat, then add more nodes in each layer.
- Whiten continuous input data (de-mean, divide by standard deviation).
- Ensure that you are performing cross-validation or evaluating against a test set holdout.
- For classification problems, apply class weights to balance labels (in the model.fit() function).
- Use RMSprop or Adam as your first optimizer.
- Check that your loss function makes sense given your label value ranges. Also, check that your output layer activation makes sense!
- Monitor accuracy as a metric for classification problems, MSE or MAE for regression.
- Start with 20 epochs, increase if the validation loss has not yet plateaued.
- Start with a batch size of 16 and then double it to see effect on model performance.

Past Project Examples

Examples of Past Student Projects That Had Some Success

- Detect defects in images of leather samples (quality assurance).
- Classify the ‘grade’ of a marble slab based on its photo (approximate value of the sample).
- Detect ‘out of control’ fires to notify emergency responders (based on CCTV images).
- Label whether images of refuse contain recyclable materials or not.
- Recommend an appropriate ‘top’ for a pair of pants, or vice versa.
- Predict the calorie content from a recipe.
- Detect whether a driver’s eyes are not on the road.
- Predict whether a review is likely to be ‘helpful’ to other consumers.

Examples of Past Student Projects That Struggled

- Predict stock price movements using textual data from Reddit posts.
- Forecast sales at Walmart locations.
- Predict engagement with a YouTube video based on its thumbnail and creator features.
- Translating sign language into written text.
- Classifying actions from sports video recordings.
- Predicting pass success from professional hockey data.

Coding Assignment

Basic Prediction Exercise

- I am providing you with a sample of real data from the Blue Bikeshare service (Boston bikeshare).
- Your goal is to use this data to predict how long an individual's bike rental / trip will last (at the time they begin the rental). See instructions for the assignment on Github.

Deliverable

- Produce a Colab notebook documenting your work. Submit the .ipynb file via Blackboard before the assignment deadline.
- Make sure you comment your code well and follow the provided template so I am clear what you were trying to do!
- Answer conceptual questions laid out in the assignment document.
- Document your use of GenAI tools (include the prompts you used and screenshot output).
- State which other students you spoke to / interacted with when brainstorming how to solve the assignment.

Start Working Now...