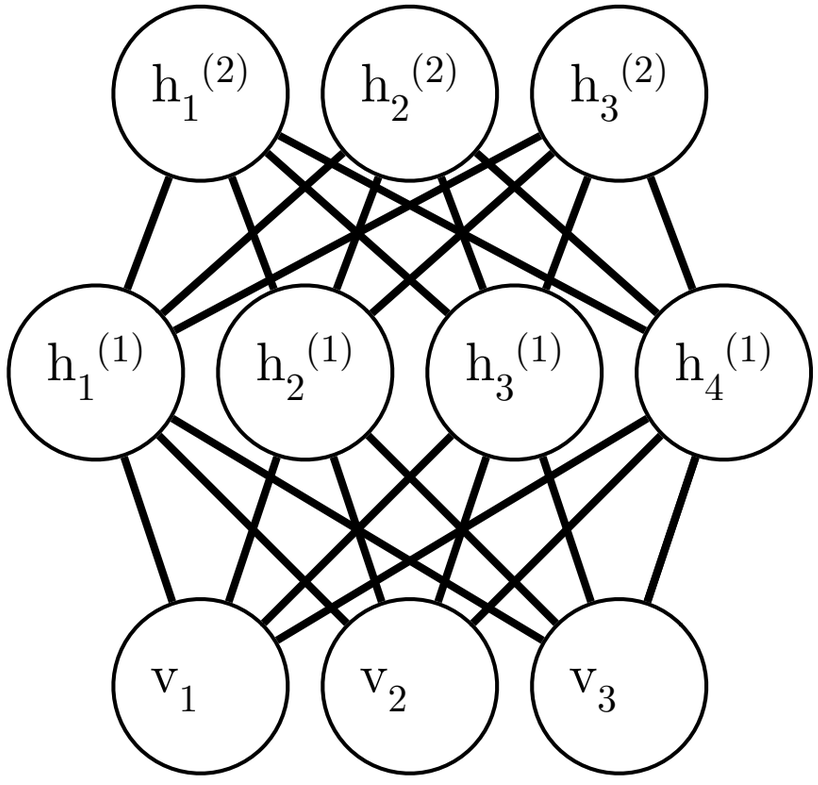
Practical Methodology for Deploying Machine Learning

Ian Goodfellow

(An homage to "Advice for Applying Machine Learning" by Andrew Ng)

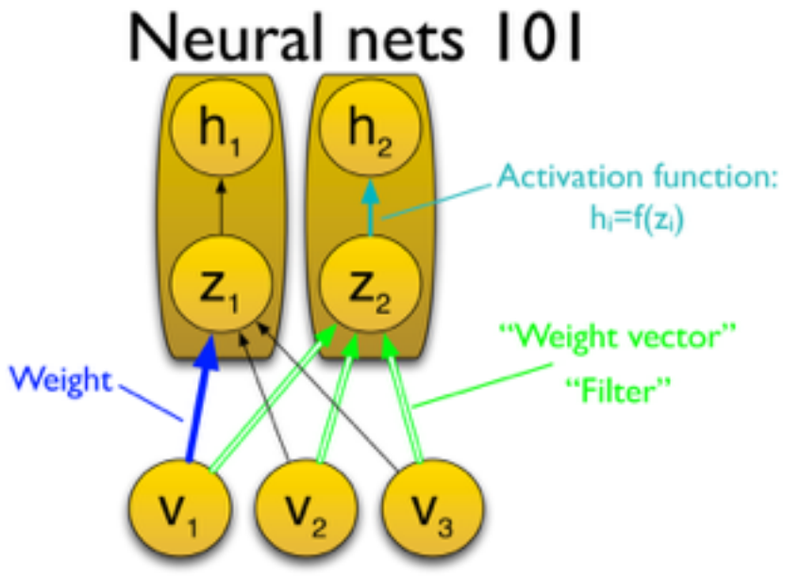
What drives success in ML?

Arcane knowledge of dozens of obscure algorithms?

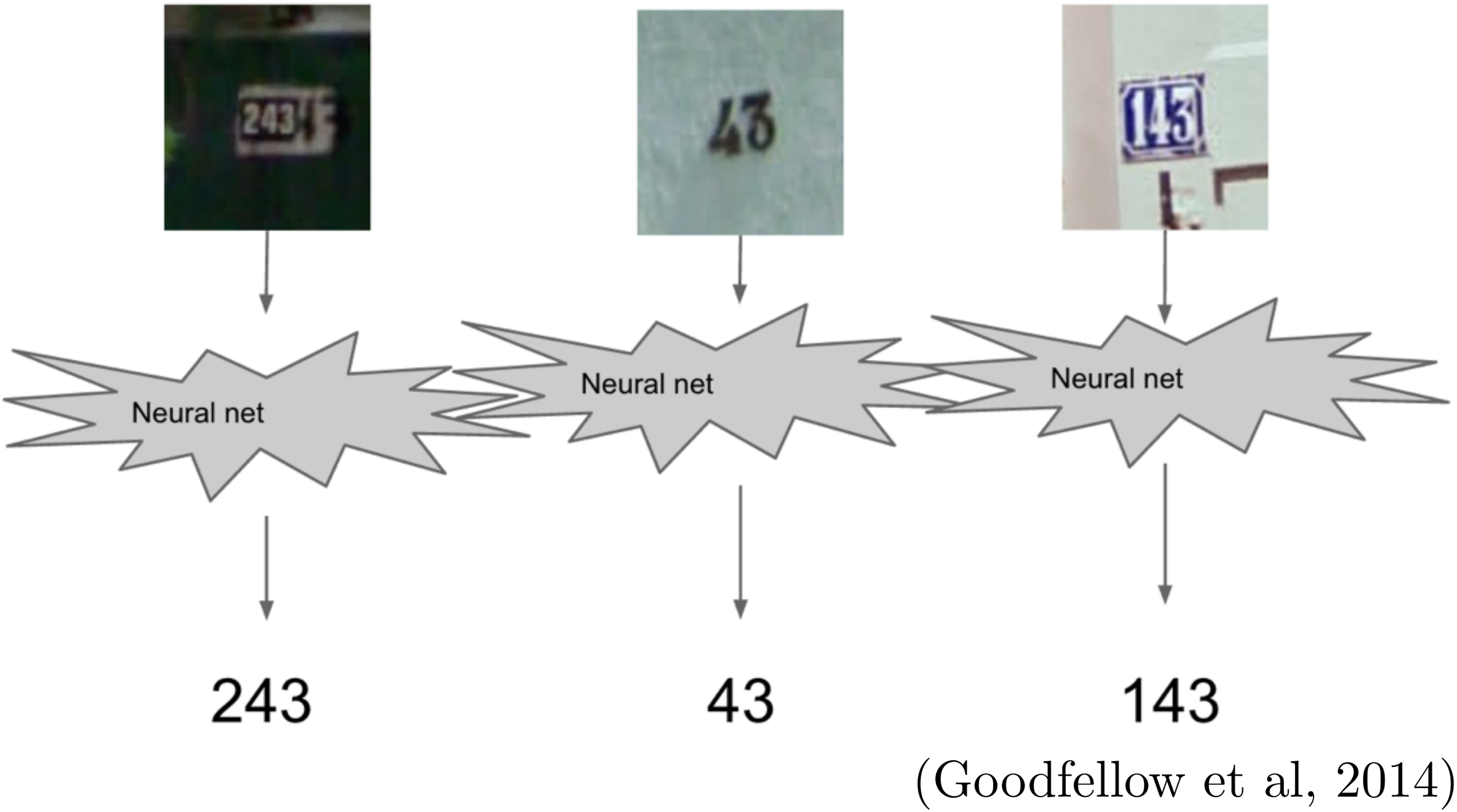


Mountains of data?

Knowing how to apply 3-4 standard techniques?



Street View Transcription



(Goodfellow et al, 2014)

3 Step Process

- Use needs to define metric-based goals
- Build an end-to-end system
- Data-driven refinement

Identify needs

- High accuracy or low accuracy?
- Surgery robot: high accuracy
- Celebrity look-a-like app: low accuracy

Choose Metrics

- Accuracy? (% of examples correct)
- Coverage? (% of examples processed)
- Precision? (% of detections that are right)
- Recall? (% of objects detected)
- Amount of error? (For regression problems)

End-to-end system

- Get up and running ASAP
- Build the simplest viable system first
- What baseline to start with though?
  - Copy state-of-the-art from related publication

Deep or not?

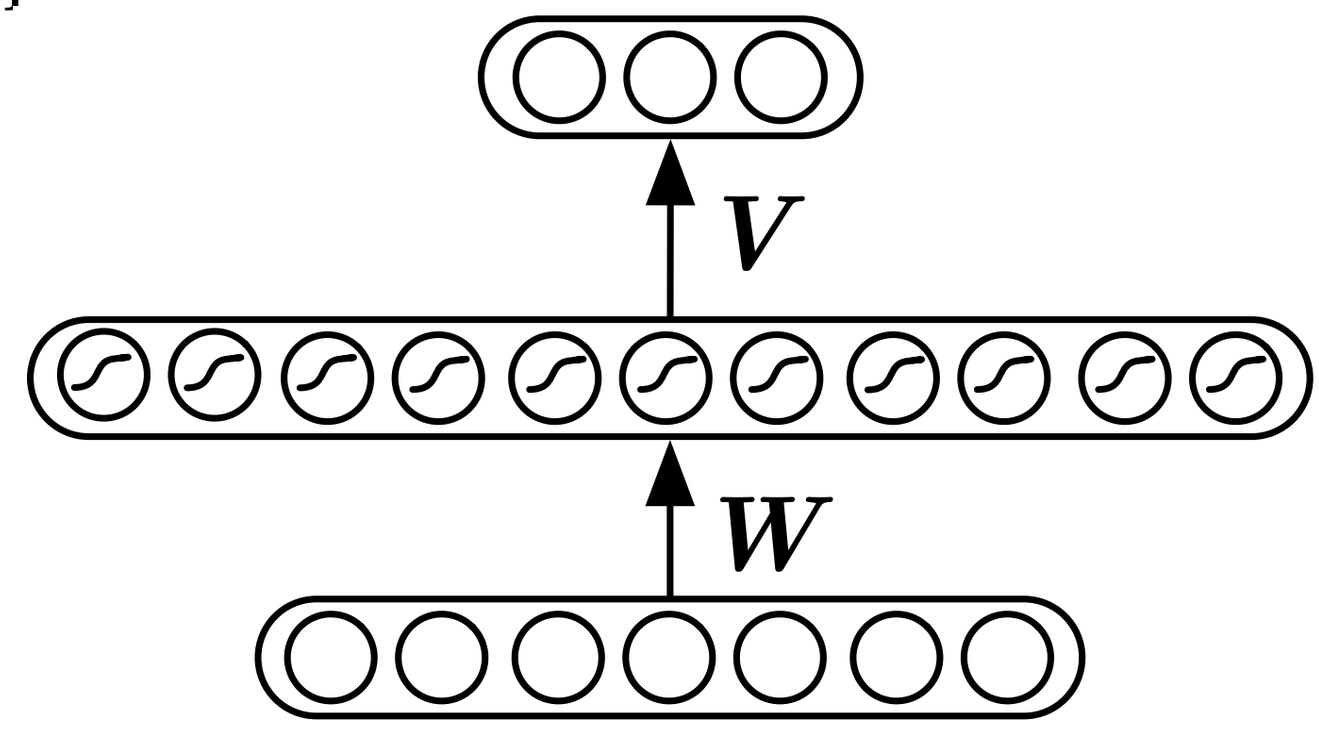
- Lots of noise, little structure -> not deep
- Little noise, complex structure -> deep
- Good shallow baseline:
  - *Use what you know*
  - Logistic regression, SVM, boosted tree are all good

What kind of deep?

- No structure -> fully connected
- Spatial structure -> convolutional
- Sequential structure -> recurrent

Fully connected baseline

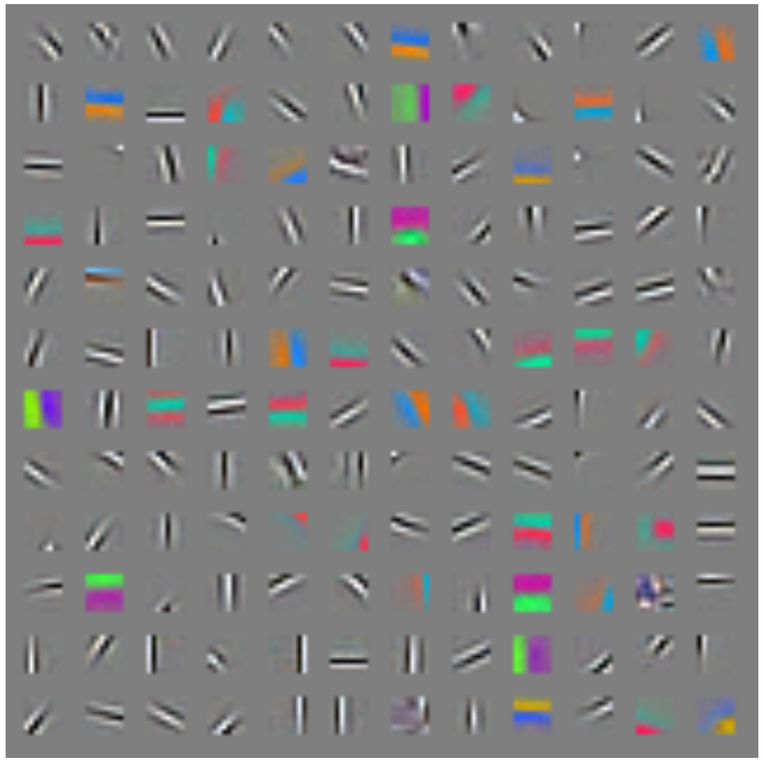
- 2-3 hidden layer feedforward network
- AKA "multilayer perceptron"
- Rectified linear units
- Dropout
- SGD + momentum



Convolutional baseline

- Inception

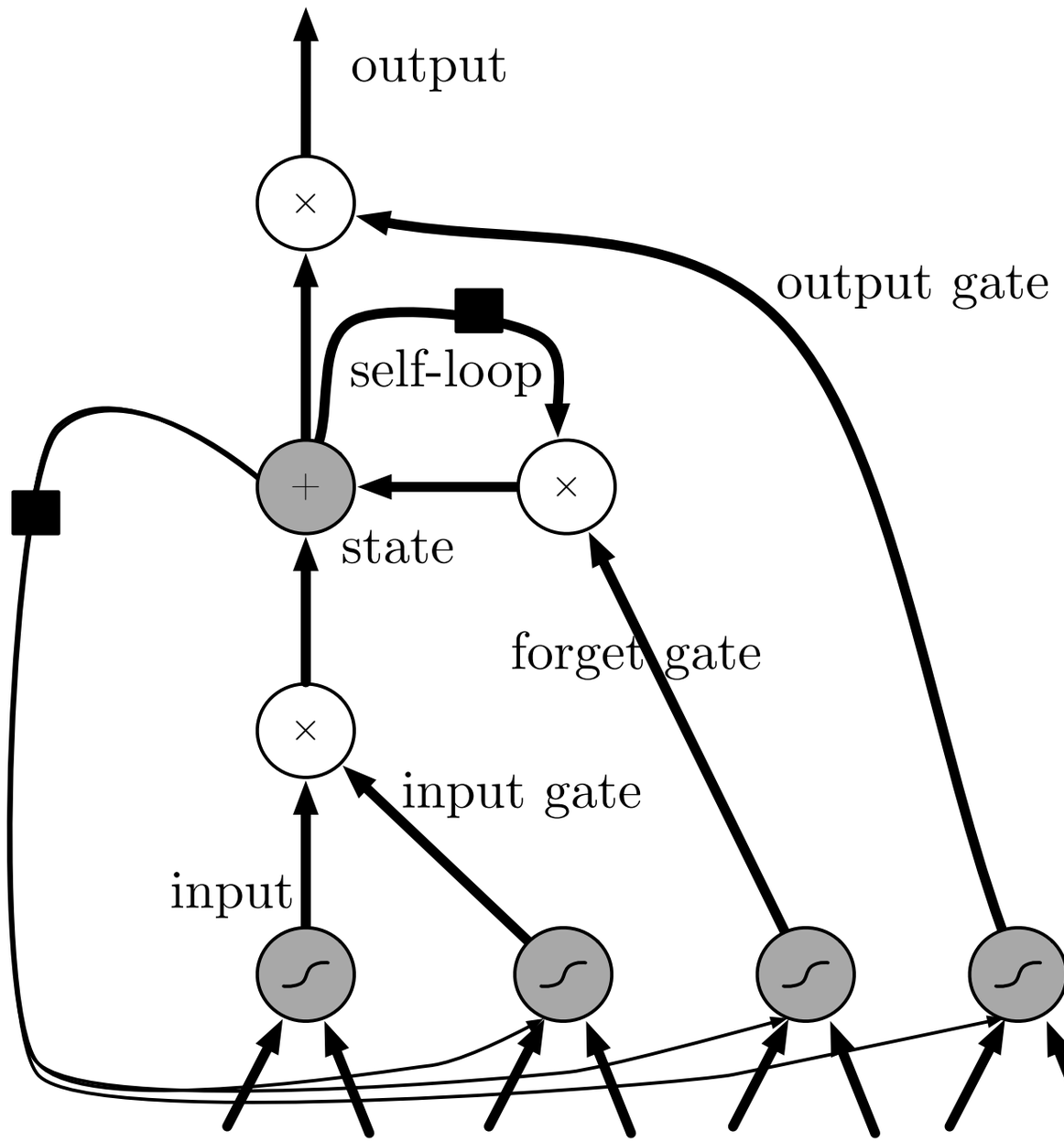
- Batch normalization



- Fallback option:
  - Rectified linear convolutional net
  - Dropout
  - SGD + momentum

Recurrent baseline

- LSTM
- SGD
- Gradient clipping
- High forget gate bias



Data driven adaptation

- Choose what to do based on data
- Don't believe hype
- Measure train and test error
  - "Overfitting" versus "underfitting"

High train error

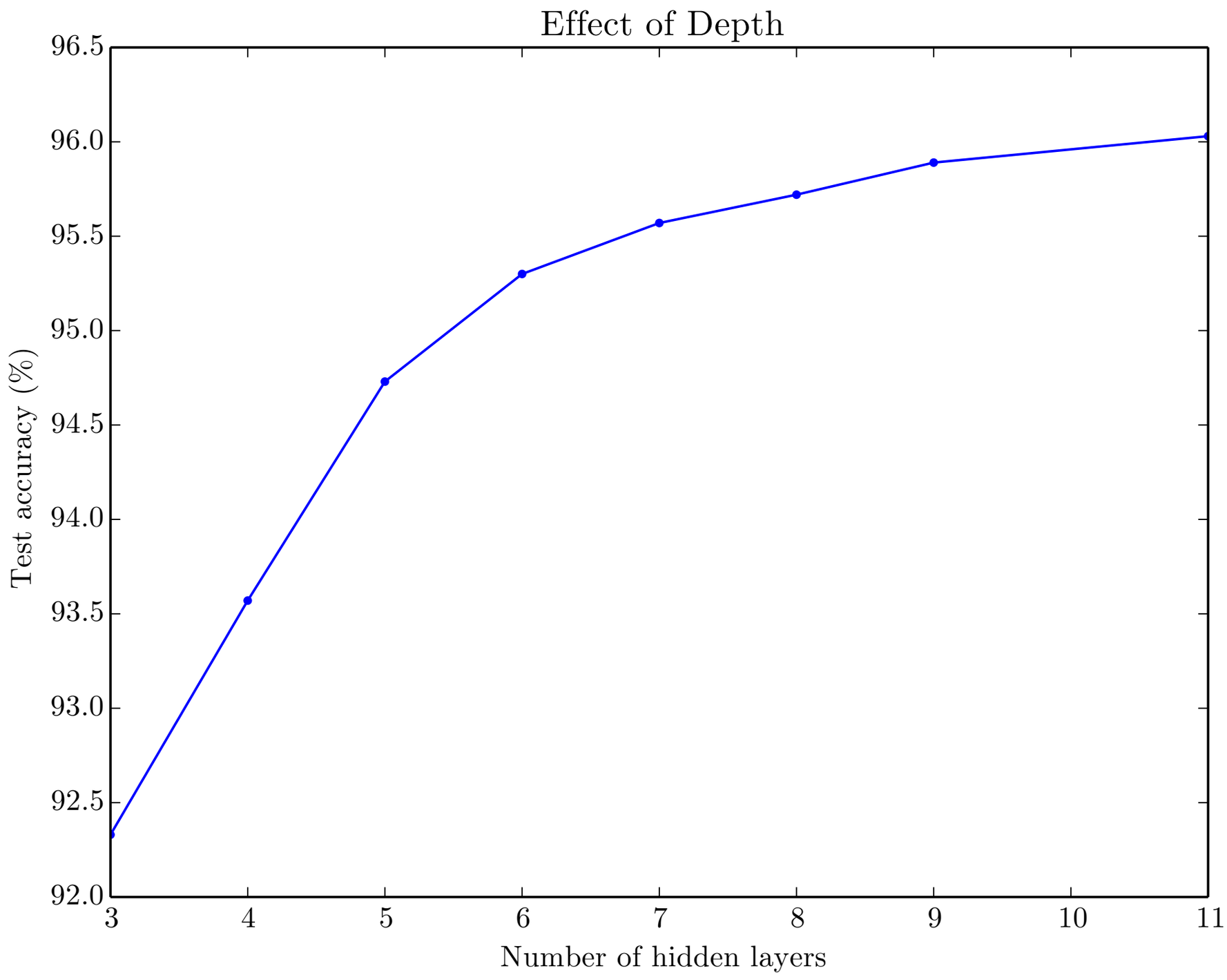
- Inspect data for defects
- Inspect software for bugs
    - Don't roll your own unless you know what you're doing
- Tune learning rate (and other optimization settings)
- Make model bigger

Checking data for defects

- Can a human process it?

26624

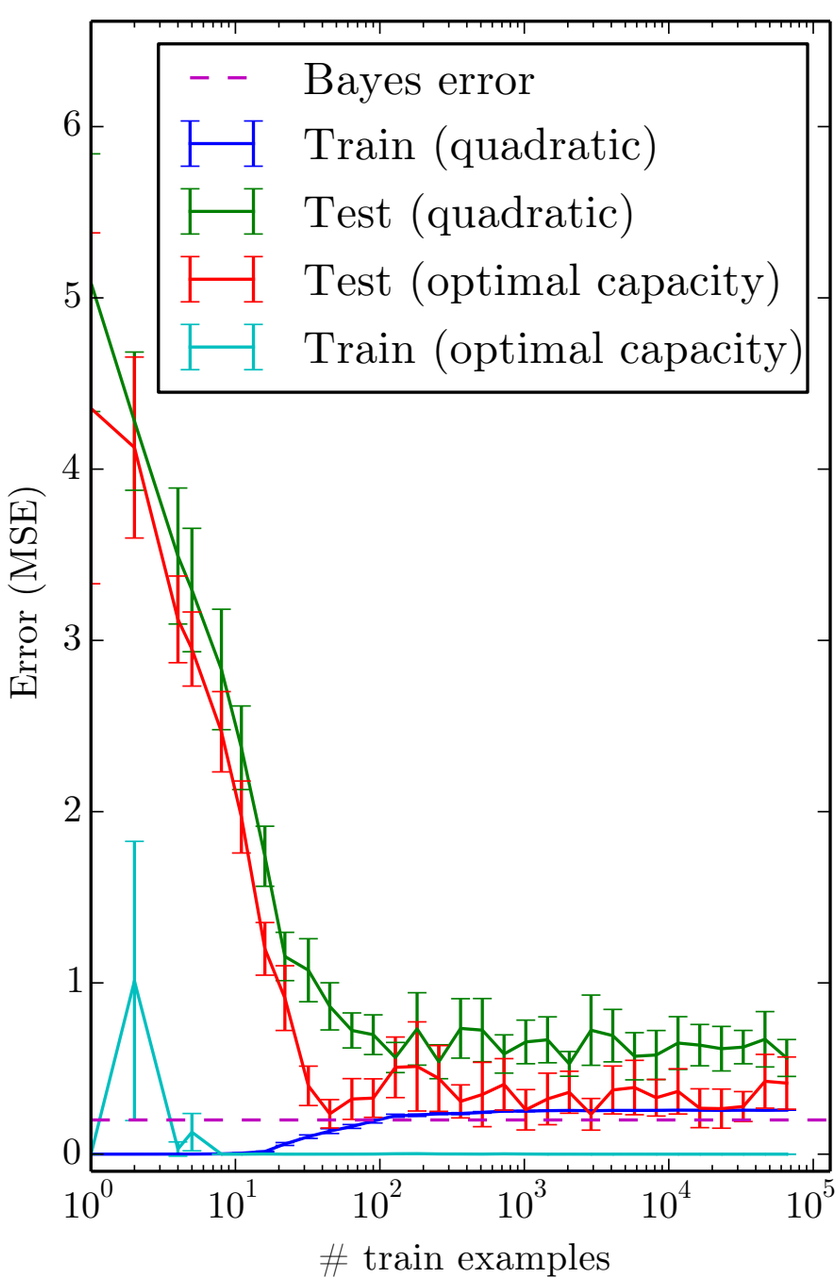
Increasing depth

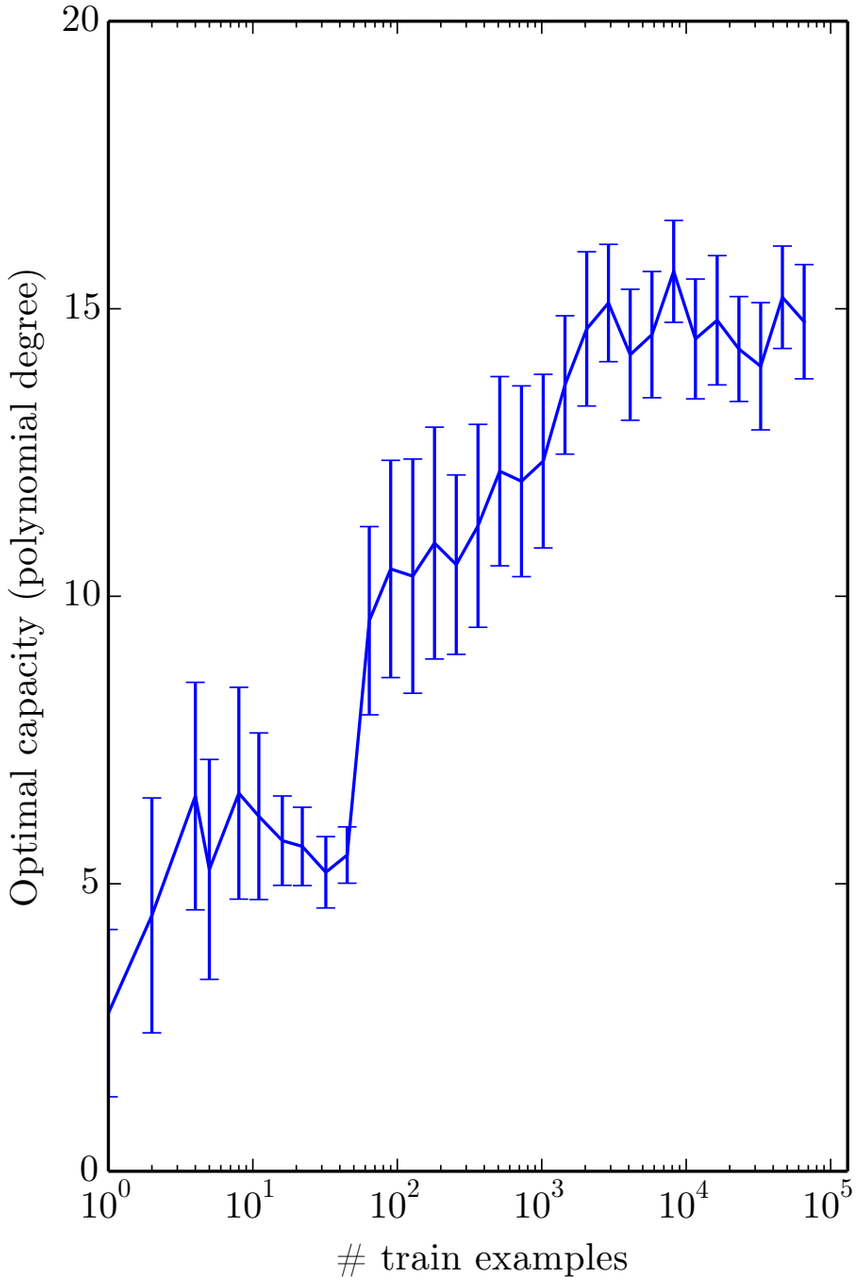


High test error

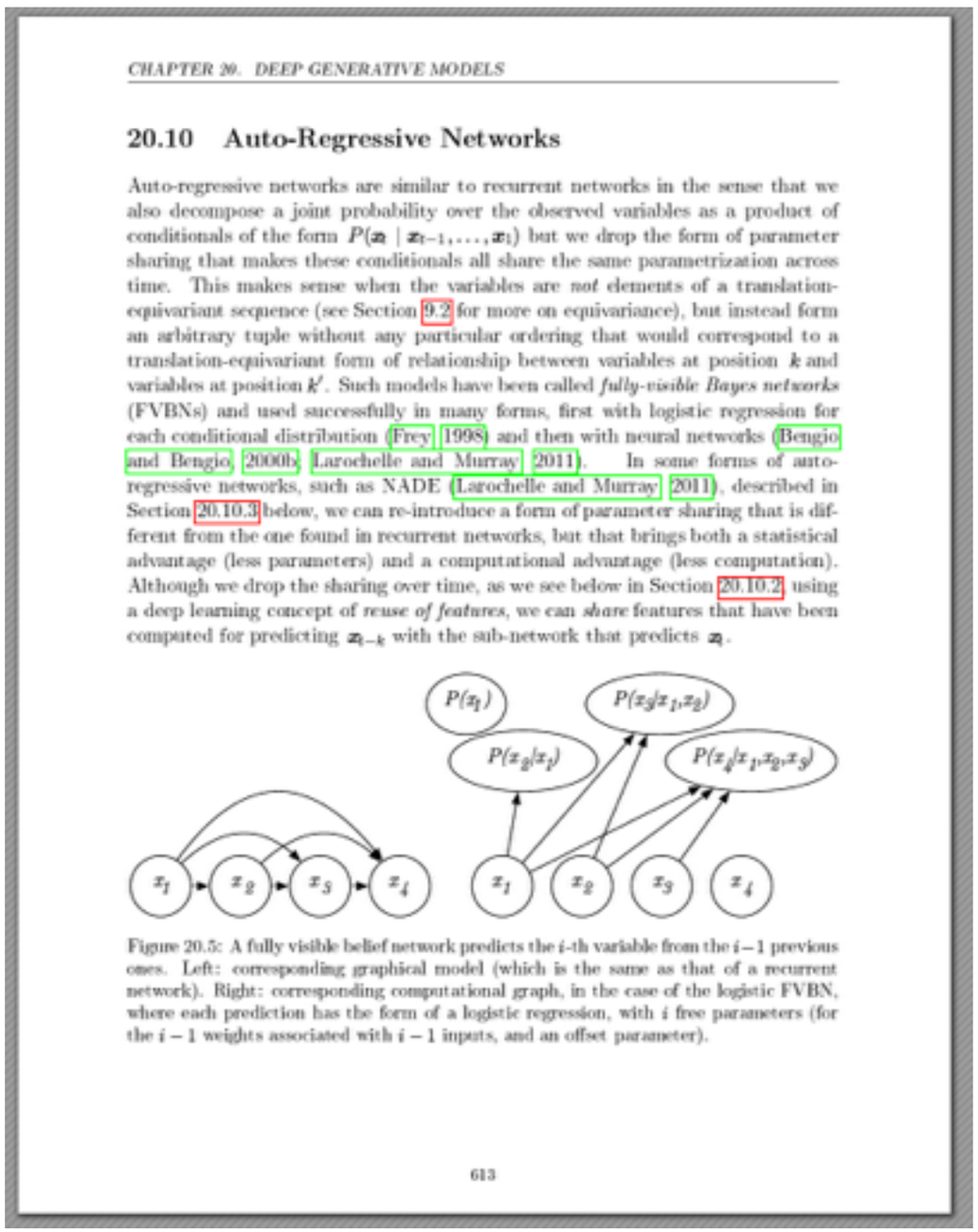
- Add dataset augmentation
- Add dropout
- Collect more data

Increasing training set size





Deep Learning textbook



CHAPTER 20. DEEP GENERATIVE MODELS

20.10 Auto-Regressive Networks

Auto-regressive networks are similar to recurrent networks in the sense that we also decompose a joint probability over the observed variables as a product of conditionals of the form $P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_1)$ but we drop the form of parameter sharing that makes these conditionals all share the same parametrization across time. This makes sense when the variables are *not* elements of a translation-equivariant sequence (see Section 9.2 for more on equivariance), but instead form an arbitrary tuple without any particular ordering that would correspond to a translation-equivariant form of relationship between variables at position k and variables at position k'. Such models have been called *fully-visible Bayes networks* (FVBNs) and used successfully in many forms, first with logistic regression for each conditional distribution (Frey, 1998) and then with neural networks (Bengio and Bengio, 2000b; Larochelle and Murray, 2011). In some forms of auto-regressive networks, such as NADE (Larochelle and Murray, 2011), described in Section 20.10.3 below, we can re-introduce a form of parameter sharing that is different from the one found in recurrent networks, but that brings both a statistical advantage (less parameters) and a computational advantage (less computation). Although we drop the sharing over time, as we see below in Section 20.10.2, using a deep learning concept of *reuse of features*, we can *share* features that have been computed for predicting $\boldsymbol{x}_{t-k}$ with the sub-network that predicts $\boldsymbol{x}_t$.

Figure 20.5: A fully visible belief network predicts the i-th variable from the $i-1$ previous ones. Left: corresponding graphical model (which is the same as that of a recurrent network). Right: corresponding computational graph, in the case of the logistic FVBN, where each prediction has the form of a logistic regression, with i free parameters (for the $i-1$ weights associated with $i-1$ inputs, and an offset parameter).

613

Yoshua Bengio

Ian Goodfellow

Aaron Courville

goodfeli.github.io/dlbook