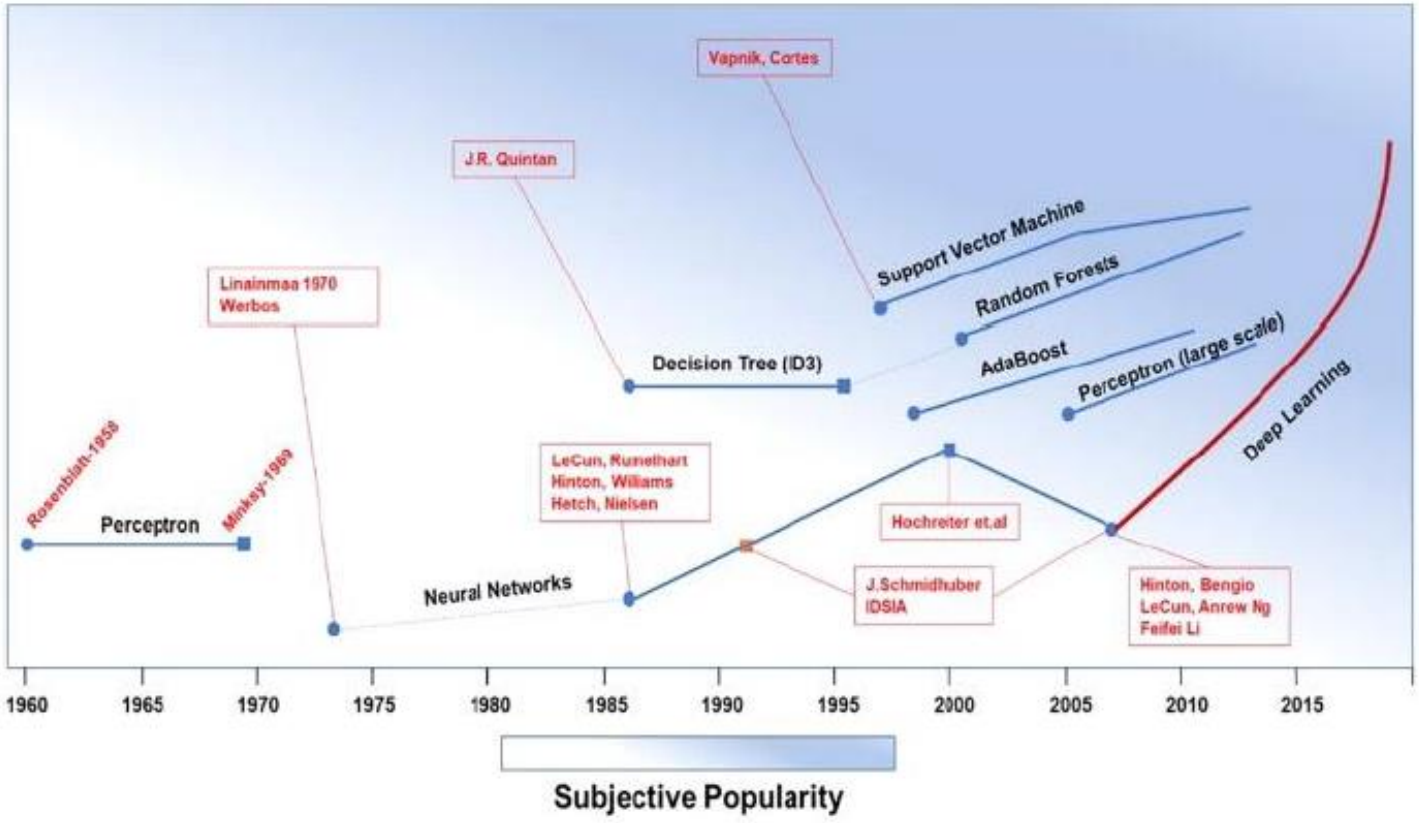
Topic 9. Deep Learning

Historical Evolution of Deep Learning

- Neural network (NN) appeared in late 1980s.
- NN resurfaced around 2010 with the new name deep learning.



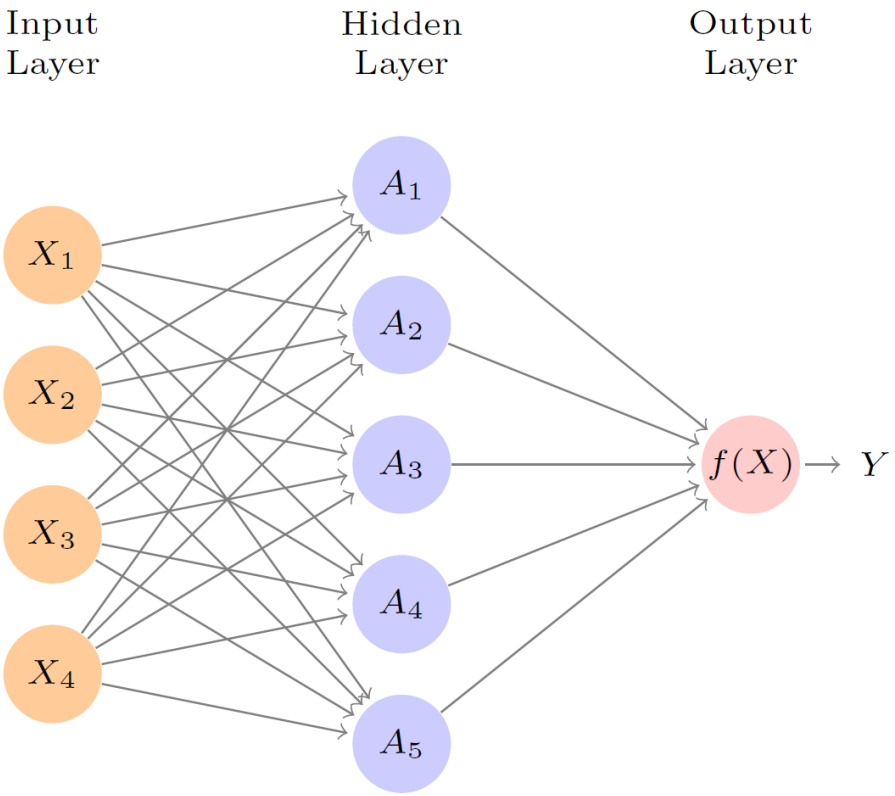
"Review of Machine Learning and Deep Learning Application in Mine Microseismic Event Classification", *Mining of Mineral Deposits*, 15:19-26.

Outline

- Basics: single layer neural network and fitting
- More complex neural networks
- When to use deep learning

Basics:
Single Neural Network and Fitting

Single Layer Neural Networks



$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

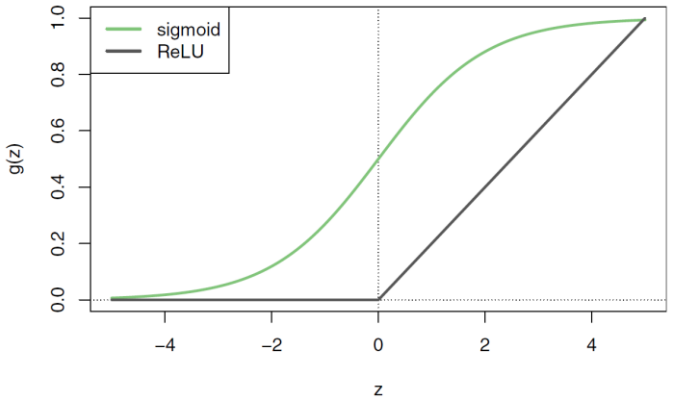
$$= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

Hidden Layer

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

- A_k is a transformation of the original features (predictors)
- $g(z)$ is a nonlinear activation function specified in advance
  - Sigmoid activation function

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$



  - ReLU (rectified linear unit) activation function

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise.} \end{cases}$$

Output Layer

- $f(X)$ is a linear function of activations

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

Neurons in Human Brain

- The name neural network originally derived from thinking of hidden units as analogous to neurons in brain.
- Activations A_k close to 1 are *firing* and close to 0 are *silent*

(dana.org)

Fitting a Neural Network

- A nonlinear least squares problem

$$\underset{\{w_k\}_1^K,\,\beta}{\text{minimize}}\ \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2,$$

where

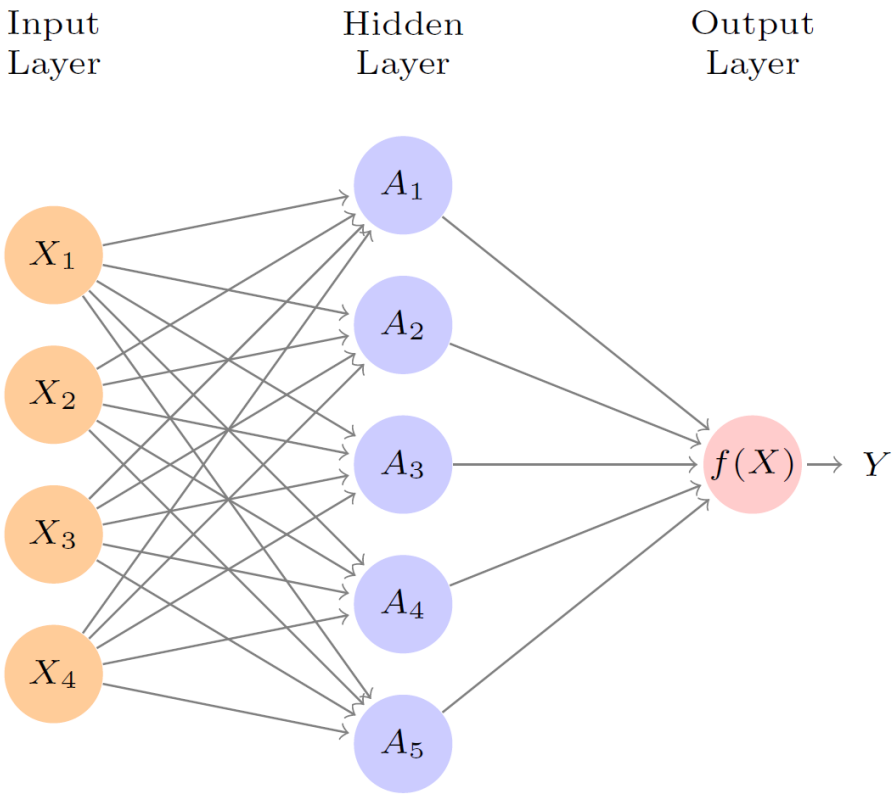
$$f(x_i) = \beta_0 + \sum_{k=1}^{K}\beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj}x_{ij}\right)$$

Total Number of Parameters

- $p = \#inputs$, $K = \#activations(nodes\ in\ the\ hidden\ layer)$

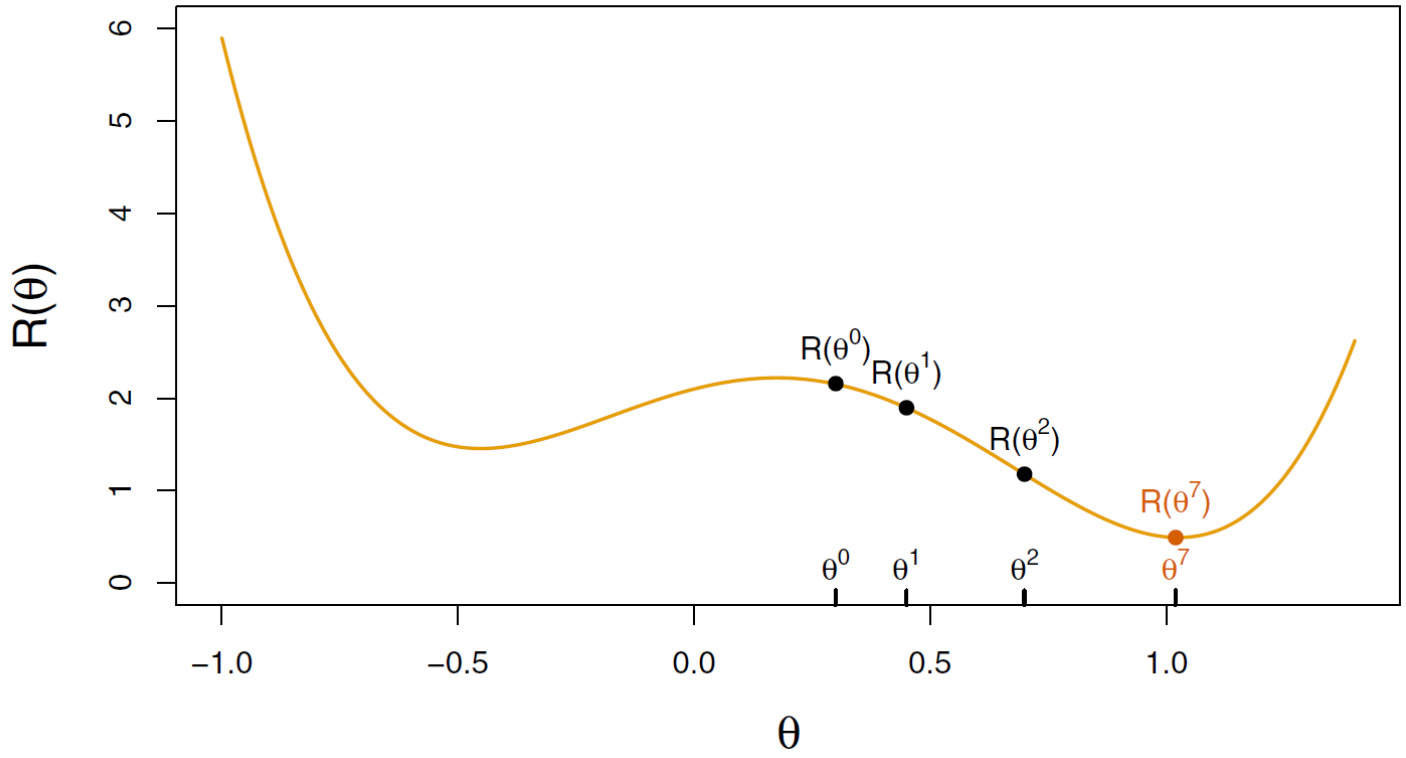
  Number of parameters $= (p+1) \times K + (K+1) = pK + 2K + 1$

  Example: $p = 4, K = 5 \Rightarrow pK + 2K + 1 = 31$



Issues in Fitting and Solution

- Nonconvex problem: local vs. global minimum
- Slow learning using gradient descent

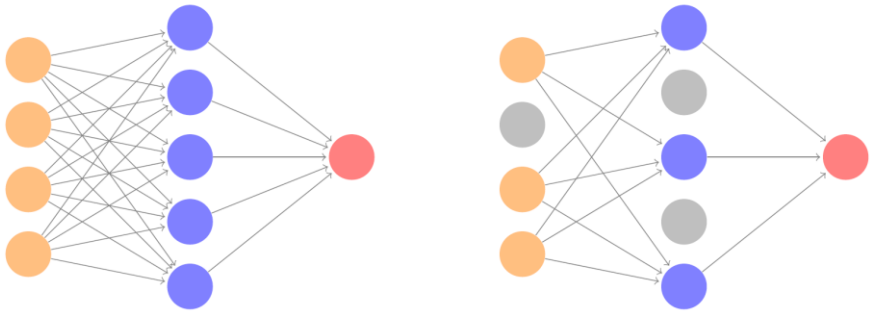


Issues in Fitting and Solution

- Large number of parameters: overfitting
- Regularization
  - Ridge or Lasso regularization

$$R(\theta;\lambda) = -\sum_{i=1}^{n}\sum_{m=0}^{9} y_{im}\log(f_m(x_i)) + \lambda\sum_{j}\theta_j^2$$

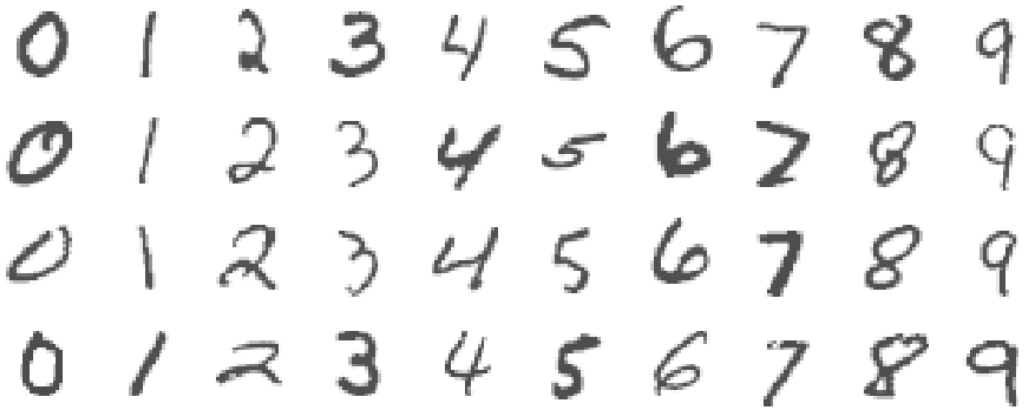
  - Dropout learning: randomly remove a fraction of units

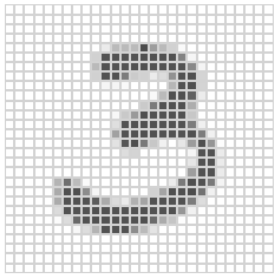


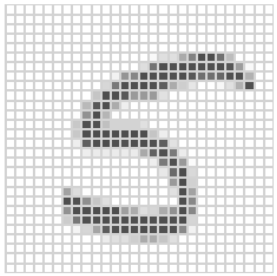
More Complex Neural Networks

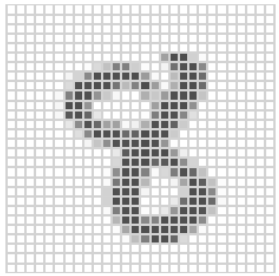
1. Multilayer Neural Network

- Handwritten digit recognition problem





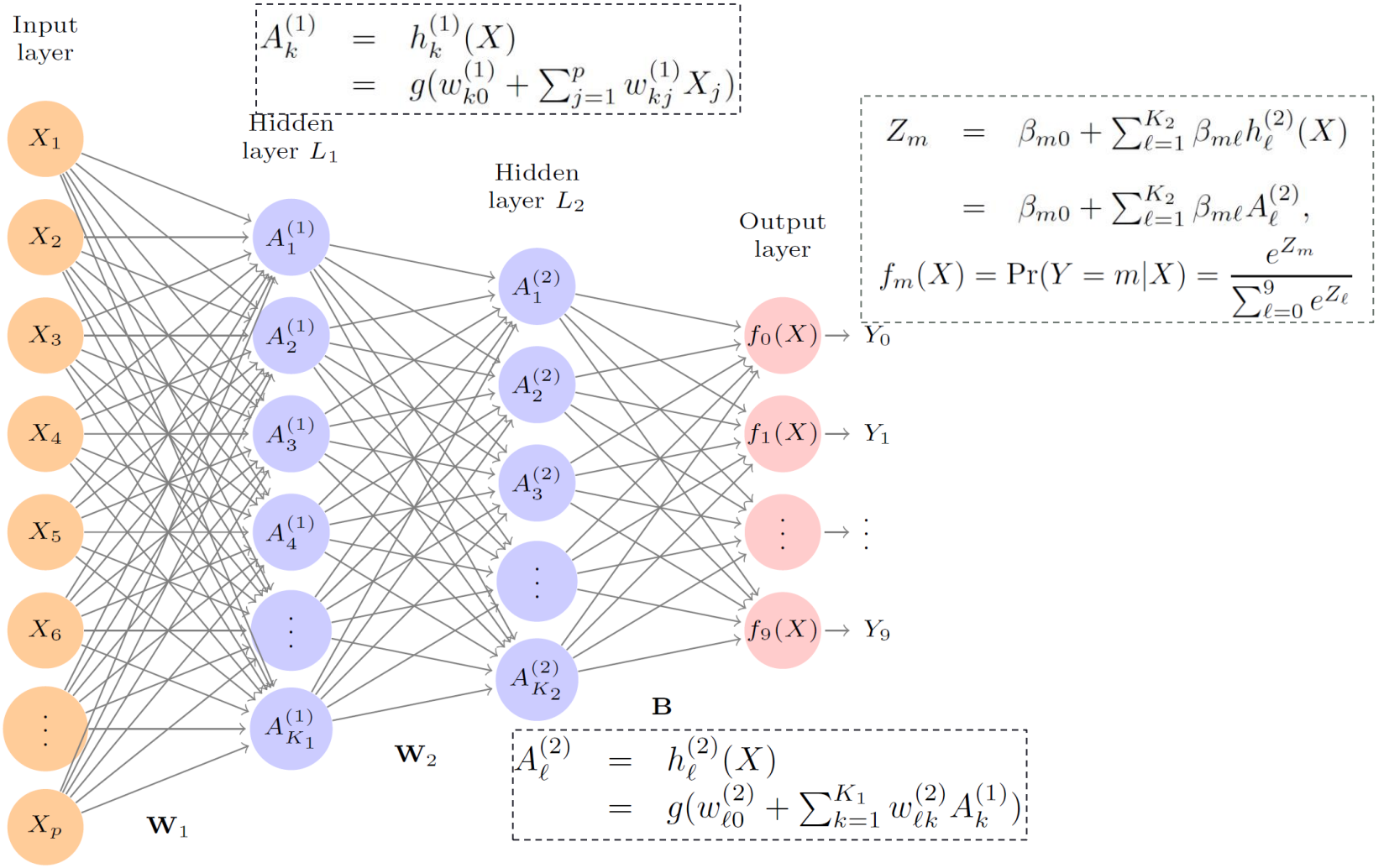




Every image has $p = 28 \times 28 = 784$ pixels.
Each pixel has a number (0-255) to represent its darkness.
Output $Y = (Y_0, Y_1, \ldots, Y_9)$ where $Y_i = 1, Y_{j,j \neq i} = 0$ if class label = i.

Neural Network with Two Hidden Layers

- L_1 has 256 units, L_2 has 128 units



Number of Parameters

- L_1: $785 \times 256 = 200{,}960$
- L_2: $257 \times 128 = 32{,}896$
- Output layer: $129 \times 10 = 1290$
- Total = 235,146

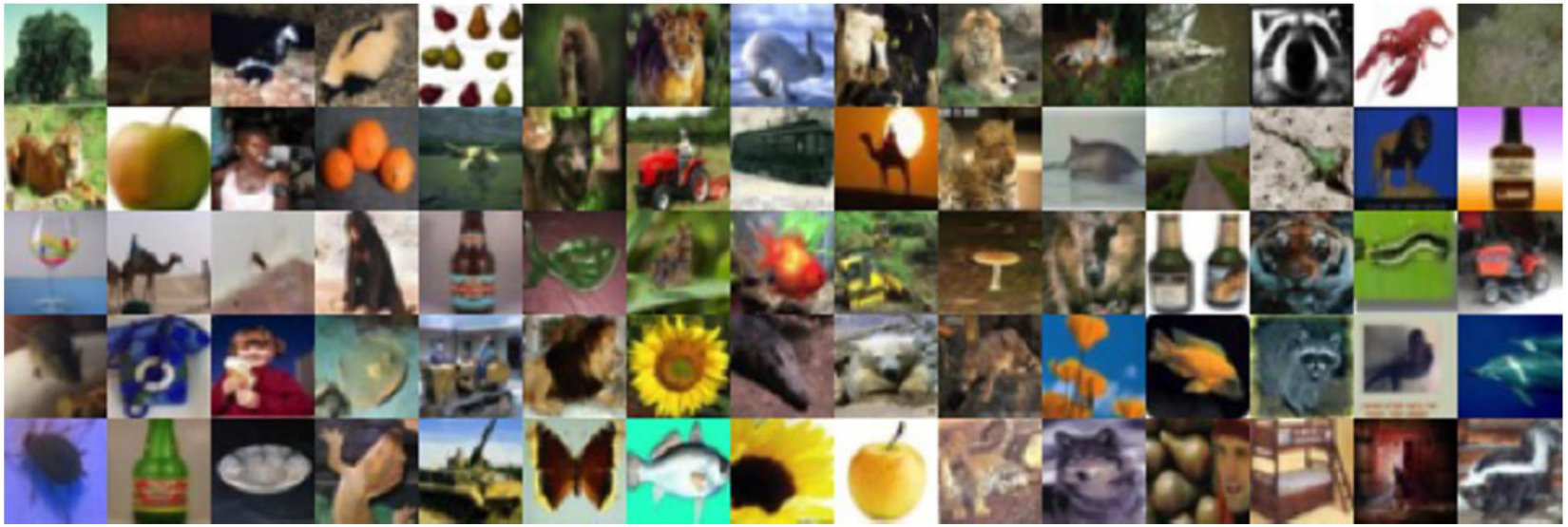
Performance Comparison

- **MNIST** data: 60,000 training images, 10,000 test images
- Prediction performance

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

2. Convolutional Neural Network (CNN)

- Image classification problem

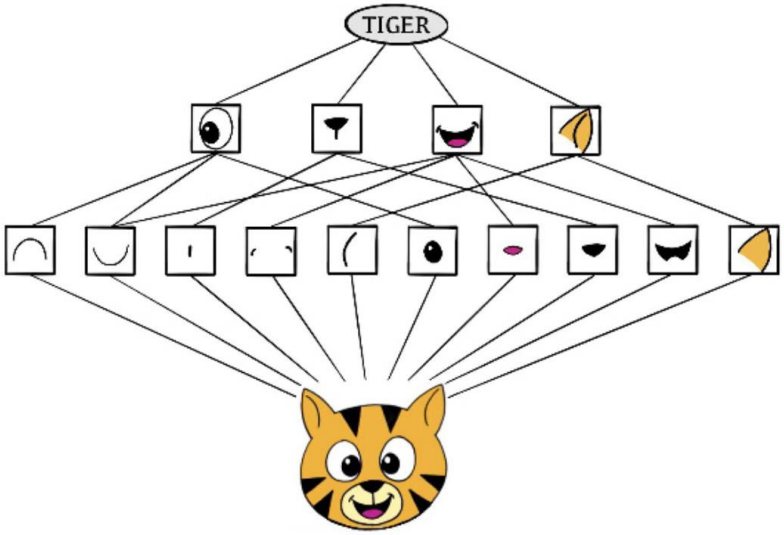


Every image has $p = 32 \times 32 = 1024$ pixels.
Each pixel has 3 numbers (0-255) to represent red, green and blue.
Output $Y = (Y_0, Y_1, \ldots)$ where $Y_i = 1, Y_{j,j \neq i} = 0$ if class label = i.

CNN

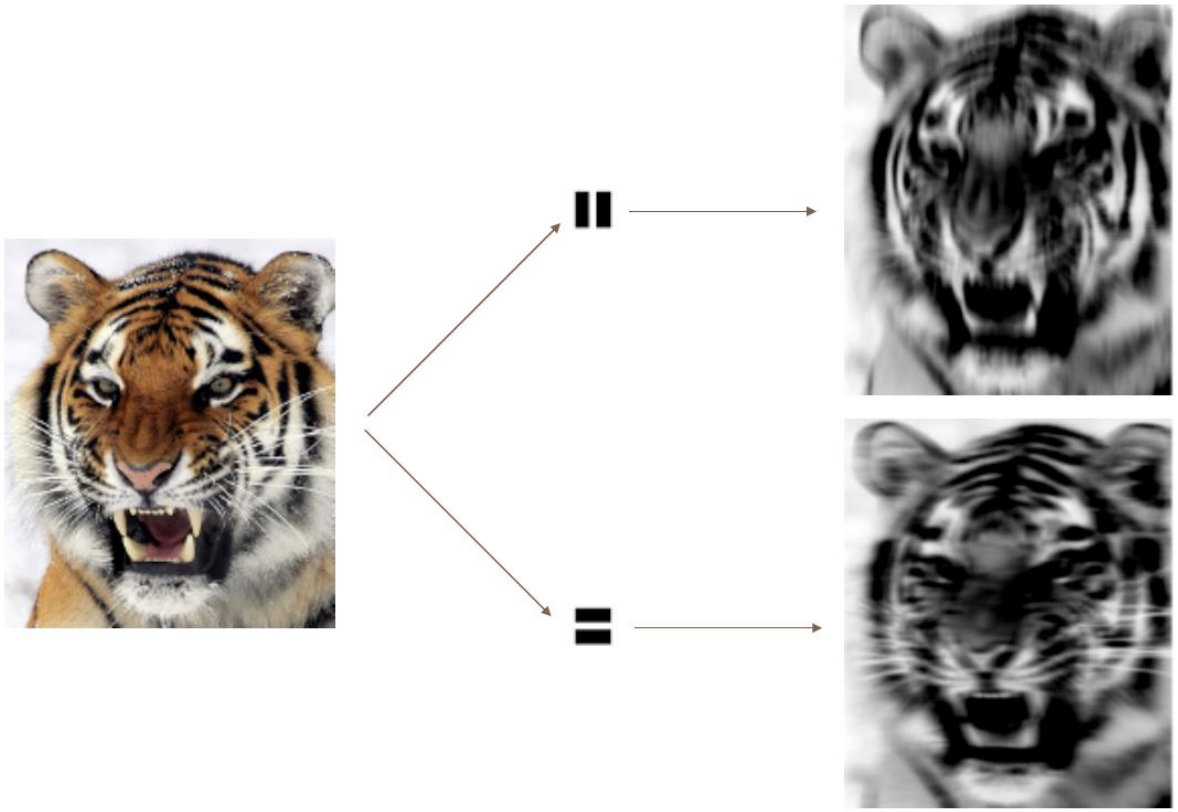
- CNNs mimic how humans classify images, by recognizing specific features or patterns anywhere in the image that distinguish each particular object class.
- (1) Identify low-level features in the input image, e.g., small edges, patches of color. (2) Combine them to form higher-level features, e.g., parts of ears and eyes. (3) Presence or absence of these features contributes to the probability of given class.

(baamboozle.com)



Two Types of Hidden Layers

- **Convolution layers**: search for small, local patterns in the image using convolution filers



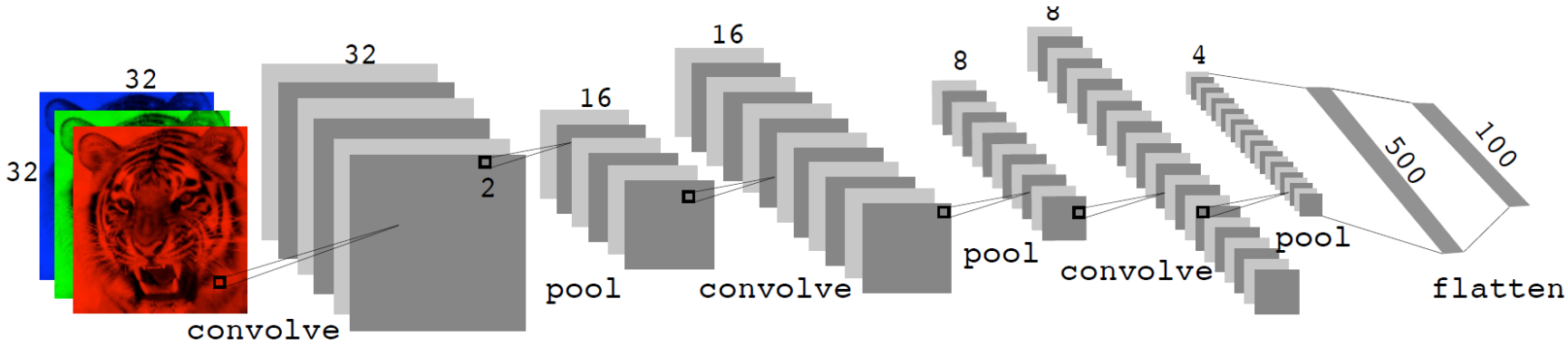
Two Types of Hidden Layers (Cont.)

- Pooling layers: downsample the small patterns to select a prominent subset

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

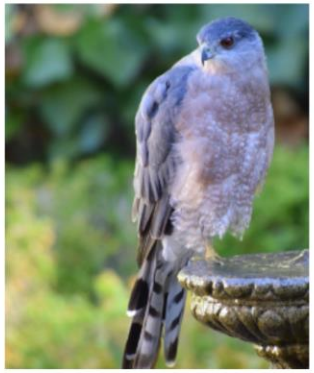
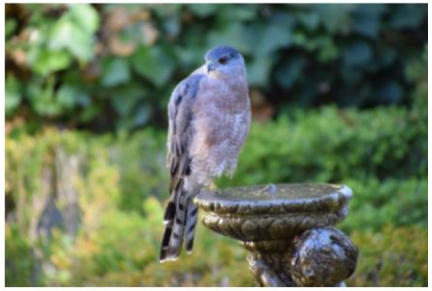
Architecture of a CNN

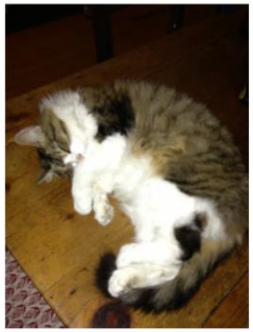
- A *convolve-then-pool* sequence



Performance

- imagenet dataset: millions of images




flamingo		Cooper's hawk		Cooper's hawk	
flamingo	0.83	kite	0.60	fountain	0.35
spoonbill	0.17	great grey owl	0.09	nail	0.12
white stork	0.00	robin	0.06	hook	0.07
Lhasa Apso		**cat**		**Cape weaver**	
Tibetan terrier	0.56	Old English sheepdog	0.82	jacamar	0.28
Lhasa	0.32	Shih-Tzu	0.04	macaw	0.12
cocker spaniel	0.03	Persian cat	0.04	robin	0.12

3. Recurrent Neural Network (RNN)

- Many data sources are sequential in nature, calling for special treatment when building predictive models.
  - Documents such as book and movie reviews, articles, tweets.
  - Time series of temperature, rainfall, wind speed, air quality.
  - Financial times series such as mark indices, trading volumes, stock and bond prices, and exchange rates.
  - Recorded speech, musical recordings, and other sound recordings.
  - Handwriting, such as doctor's notes and handwritten zip codes.

- RNNs are designed to accommodate and take advantage of the sequential nature of such data.

Example

- Document classification problem

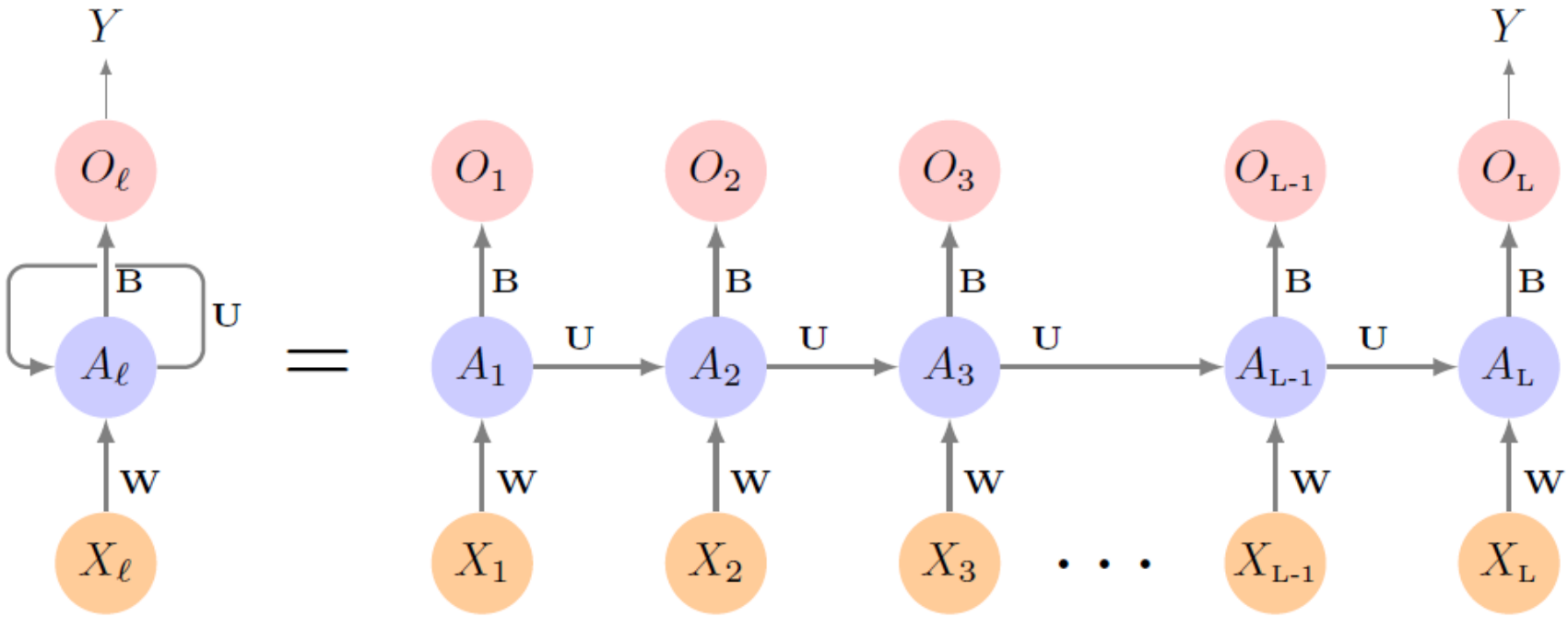
Online Movie Review

This has to be one of the worst films of the 1990s. When my friends & I were watching this film (being the target audience it was aimed at) we just sat & watched the first half an hour with our jaws touching the floor at how bad it really was. The rest of the time, everyone else in the theater just started talking to each other, leaving or generally crying into their popcorn . . .

Output/response: sentiment of the review (positive or negative)

Structure of RNN

- **Input:** $X = \{X_1, X_2, \ldots, X_L\}$ is a document with a sequence of L words.
- **Output:** often a scalar like the binary sentiment label



$$A_{\ell k} = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{\ell j} + \sum_{s=1}^{K} u_{ks} A_{\ell-1,s}\right) \qquad O_\ell = \beta_0 + \sum_{k=1}^{K} \beta_k A_{\ell k}$$

Application 1: Sentiment Analysis of Movie Reviews

- IMDb dataset: short critiques of movies
- A RNN with 32 hidden units
- Trained with dropout regularization on 25,000 reviews
- Prediction accuracy = 76%
- Using more elaborate versions (LSTM: *long term and short term* memory), performance improved to 87%

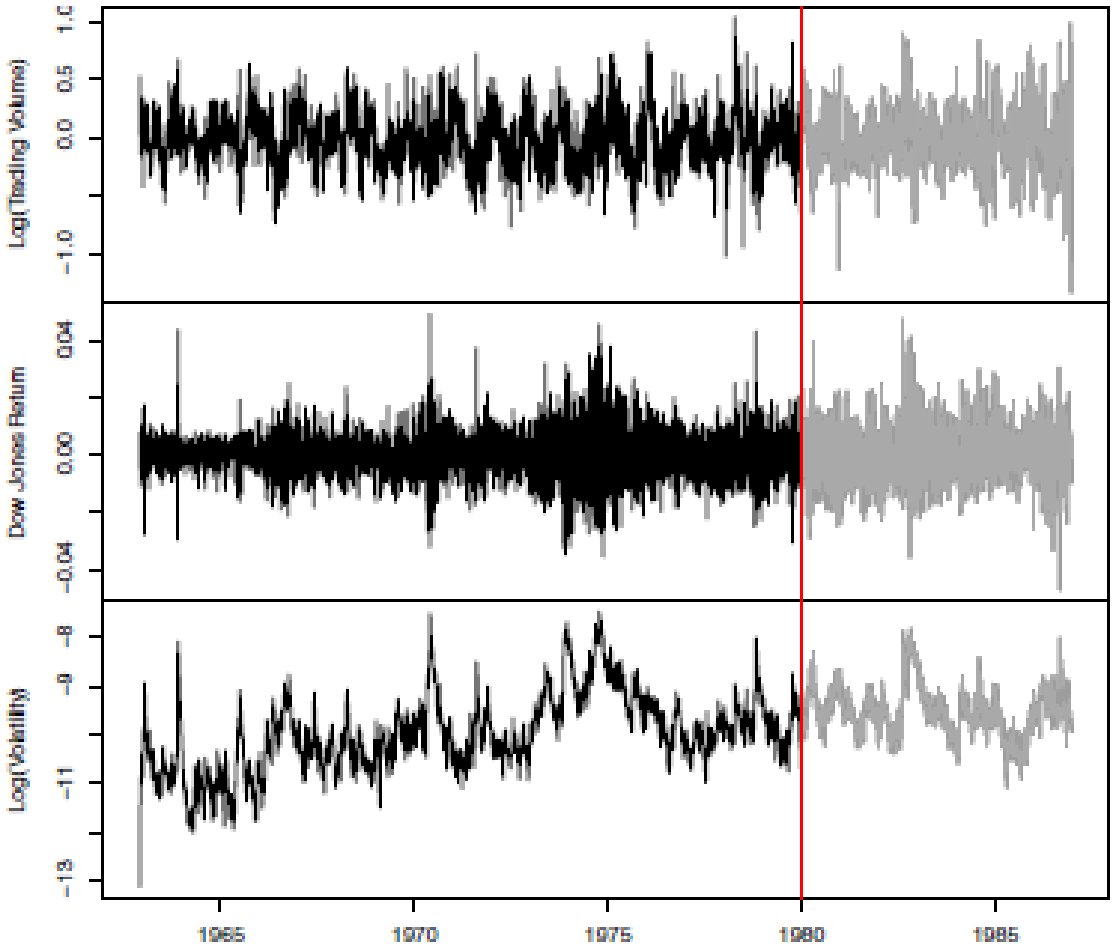
Application 2: Financial Time Series Forecasting

- **NYSE** dataset: historical trading statistics from the New York Stock Exchange during Dec 1962~Dec 1986

Log trading volume: fraction of all outstanding shares traded on a day

Dow Jones return: difference between log (Dow Jones Industrial Index) on consecutive trading days

Log volatility: absolute values of daily price movements



Prediction Problem

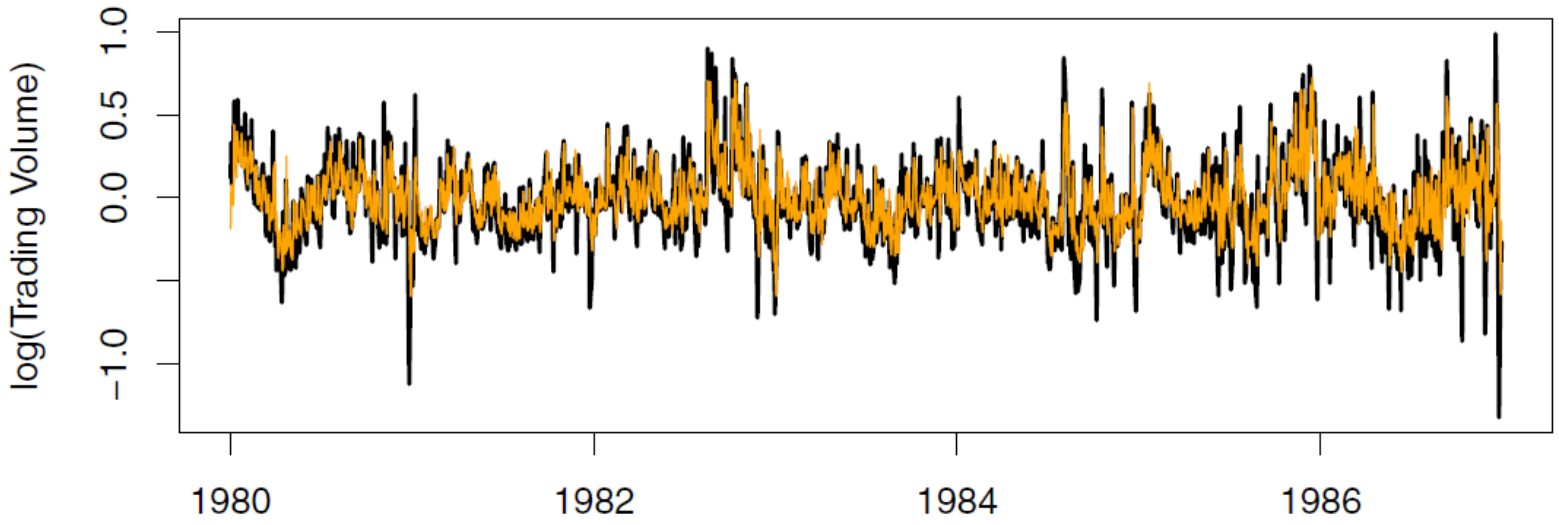
- Predicting stock price is a notoriously hard problem, but predicting **trading volume** based on recent past history is more manageable (useful for planning trading strategies).
- We wish to predict a value of trading volume v_t using
  - Past values of trading volume: $v_{t-1}, v_{t-2}, \ldots$
  - Past values of Dow Jones return: $r_{t-1}, r_{t-2}, \ldots$
  - Past values of volatility: $z_{t-1}, z_{t-2}, \ldots$
- Input $X = \{X_1, X_2, \ldots, X_L\}$ and output Y

$$X_1 = \begin{pmatrix} v_{t-L} \\ r_{t-L} \\ z_{t-L} \end{pmatrix}, \; X_2 = \begin{pmatrix} v_{t-L+1} \\ r_{t-L+1} \\ z_{t-L+1} \end{pmatrix}, \cdots, X_L = \begin{pmatrix} v_{t-1} \\ r_{t-1} \\ z_{t-1} \end{pmatrix}, \quad \text{and } Y = v_t$$

L is a predefined length (called *lag* in the context of time series)

Performance

- Training data (Dec 1962 ~ Jan 1980)
- $L = 5$
- 12 hidden units in the fitted RNN
- $R^2 = 0.42$ (The forecasted series accounts for 42% of the variance of the true data)



When to Use Deep Learning

Success Stories of Deep Learning

- CNNs in image classification: machine diagnosis of mammograms, digital X-ray images, ophthalmology eye scans, annotations of MRI scans
- RNNs in speech and language translation, forecasting, and document modeling.
- **Question:** Should we discard all the older tools, and use deep learning on every problem with data?

Model Comparison

- **Salary** dataset: predicting salary of a baseball player in 1987 using his performance statistics in 1986. $p = 19, n = 263$.
- Split into a training set (176 players) and test set (87 players)
- Comparison of three methods

Model	# Parameters	Mean Abs. Error	Test Set R^2
Linear Regression	20	254.7	0.56
Lasso	12	252.3	0.51
Neural Network	1409	257.4	0.54

- Lasso is the winner! Linear regression performs as well as neural network.

Some Guidelines

- Try both simple and complex methods
- Select a method based on *performance* vs. *complexity* tradeoff
- Deep learning is attractive when sample size of training data is extremely high, and interpretability is not a high priority.