LESSON 3: CONVOLUTIONAL AND RECURRENT NEURAL NETWORKS

Savannah Thais

Intro to Machine Learning

Princeton Wintersession 2021

01/27/2021

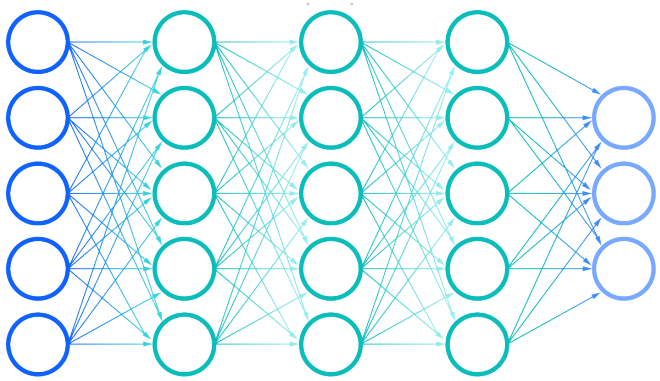
Outline for Today

- Quick Review
- Convolutional Neural Networks Theory
- Filter Coding Demo
- Recurrent Neural Network Theory
- Coding Exercises

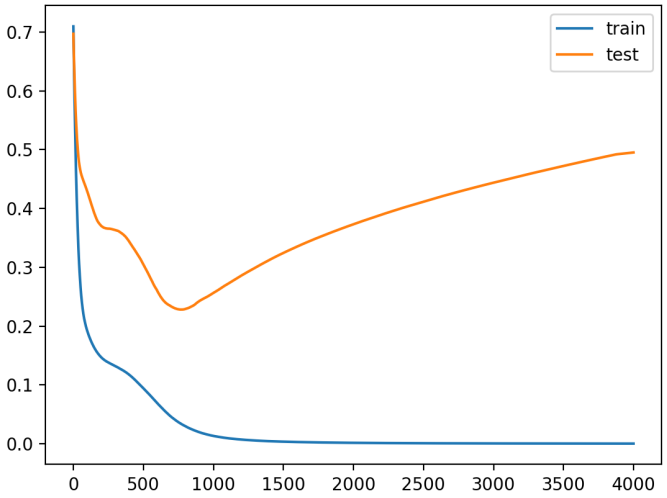
Knowledge Review

- What advantages does a NN have over a BDT?
- What is this equation?
  - What does each component represent?
  - What is an example of φ?

$$h_1 = \varphi(W_1 x + b_1)$$
$$h_2 = \varphi(W_2 h_1 + b_2)$$



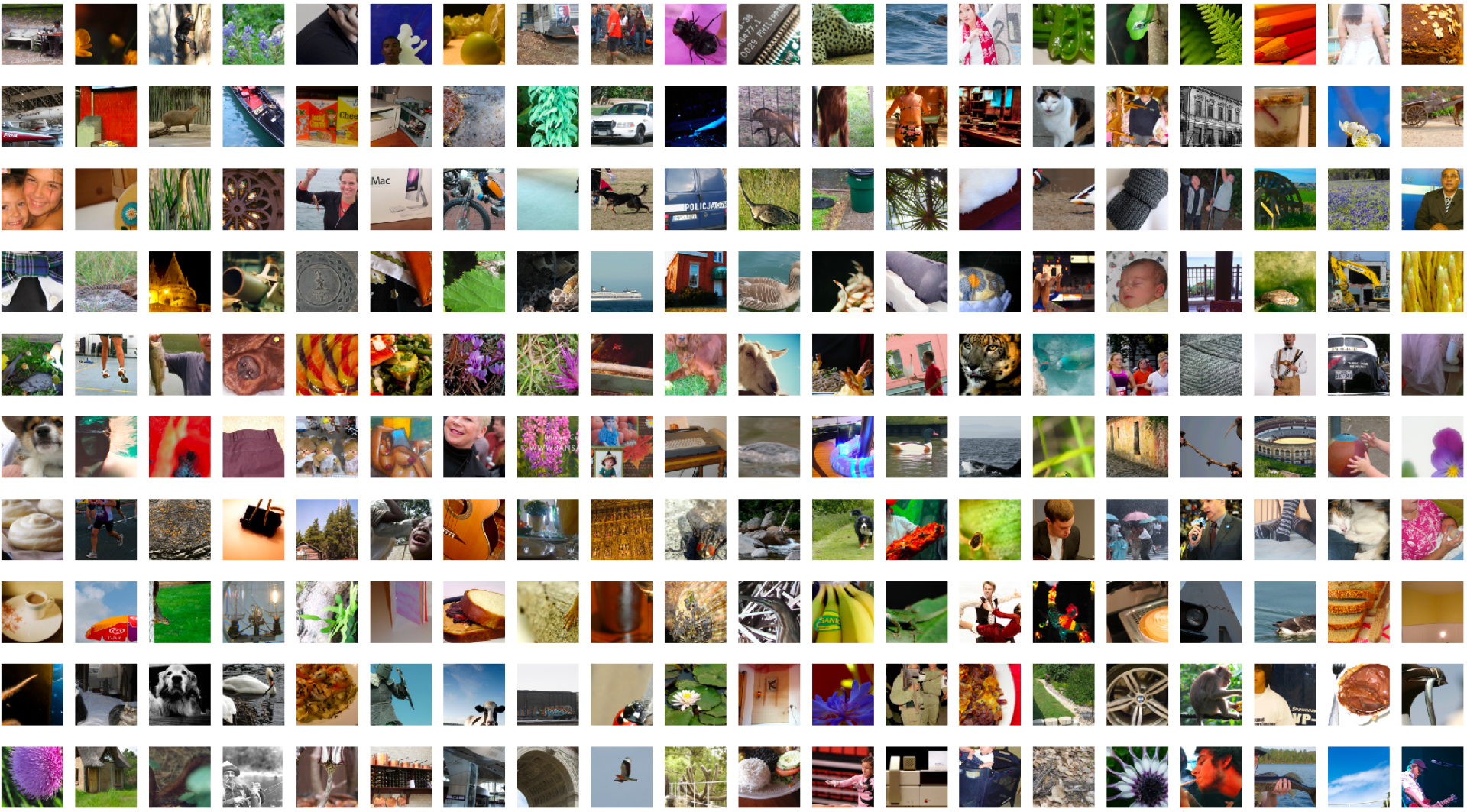
- How many hidden layers in this NN?
  - What dimensions do the input data and output labels have?
- What issue does this loss curve show?
  - If it's from a NN training, how might you fix it?
- What is gradient descent?
- What is back propagation?



Knowledge Review

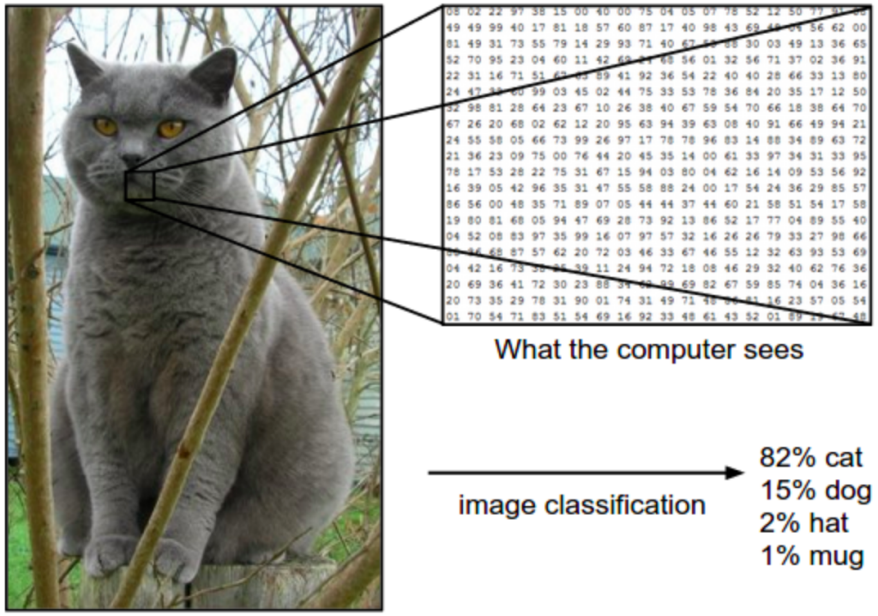
- What advantages does a NN have over a BDT?
  - Can model non-linear relationships, handle more input features, approximate any function
- What is this equation?
  - Forward propagation of a neural network
  - What does each component represent? W and b are learnable parameters (weights and biases) and φ is the non-linear activation function.
  - What is an example of φ? Softmax, ReLU, tanh, etc
- How many hidden layers in this NN?
  - 3 hidden layers, input dimension 5, output dimension 3
- What issue does this loss curve show?
  - Overtraining. You could decrease the network size, add training data, adjust hyperparameters, add regularization…
- What is gradient descent?
  - An algorithm for optimizing the loss function (move in direction of steepest descent at every training epoch).
- What is back propagation?
  - The method for propagating loss function errors through the layers of a network using the chain rule

Image Data and Convolutional Neural Networks



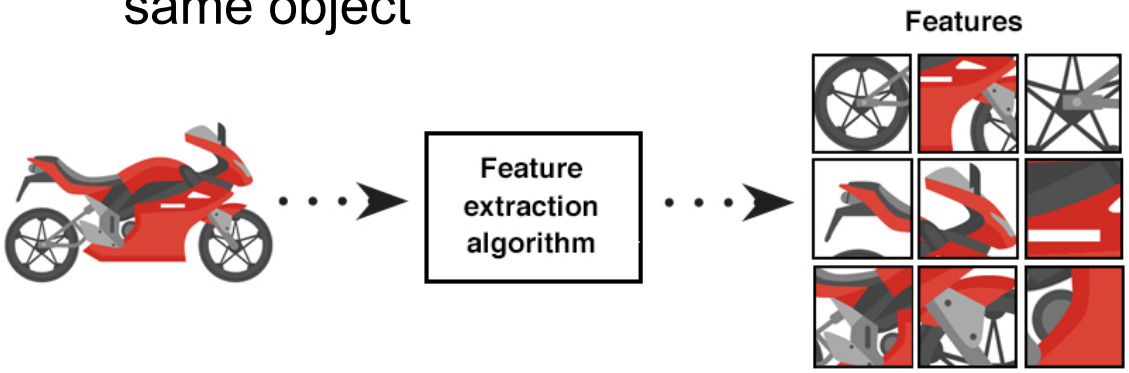
Computer Vision

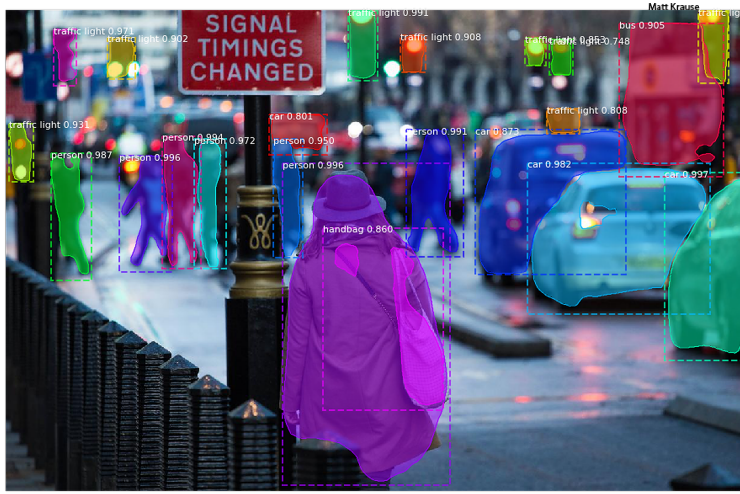
- Computer vision is a huge subfield of ML
- But computers ‘see’ images as a giant matrix of pixel values
- What are some difficulties this kind of data could present?



Computer Vision

- What are some difficulties this kind of data could present?
  - Training features: Individual pixels can be too numerous but hand engineered features are difficult to design and don't generalize well.
  - Transformations: Images with translation, rotation, or other transformations must be classified identically (invariance).
  - Complexity: images can contain several different objects or multiple instances of the same object

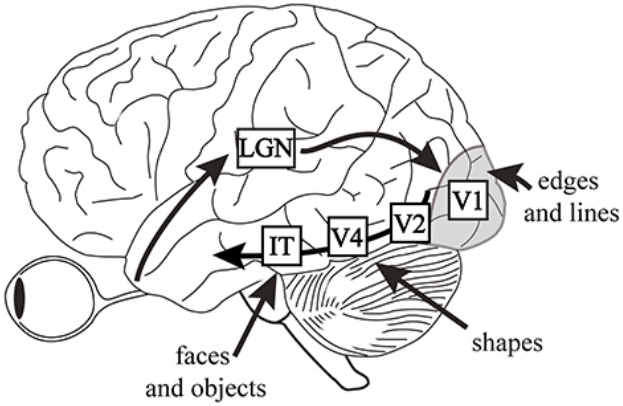


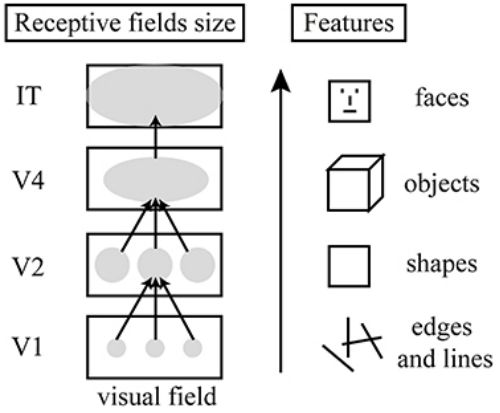


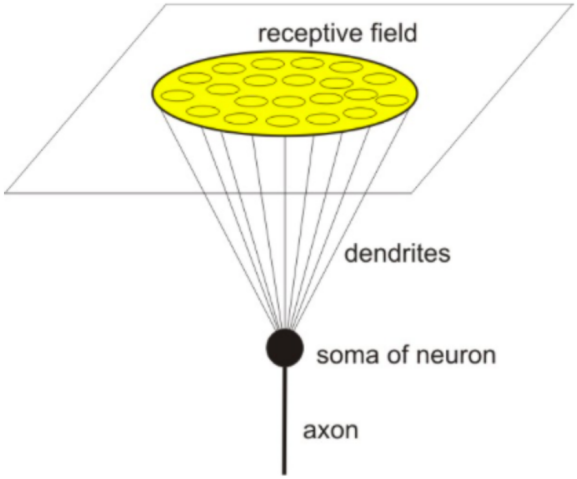
Computer Vision

As with regular NNs, researchers took inspiration from biological eyes

- Two types of cells
  - Simple cells identify basic shapes (lines) in fixed orientation
  - Complex cells respond to stimulus even when absolute position changes
- Receptive fields
  - Each neuron has a spatial area of sensitivity and these areas overlap
- Hierarchical cortex structure
  - Different visual cortexes process different information (spatial location, frequency, size/color/shape, motion, forms, etc)

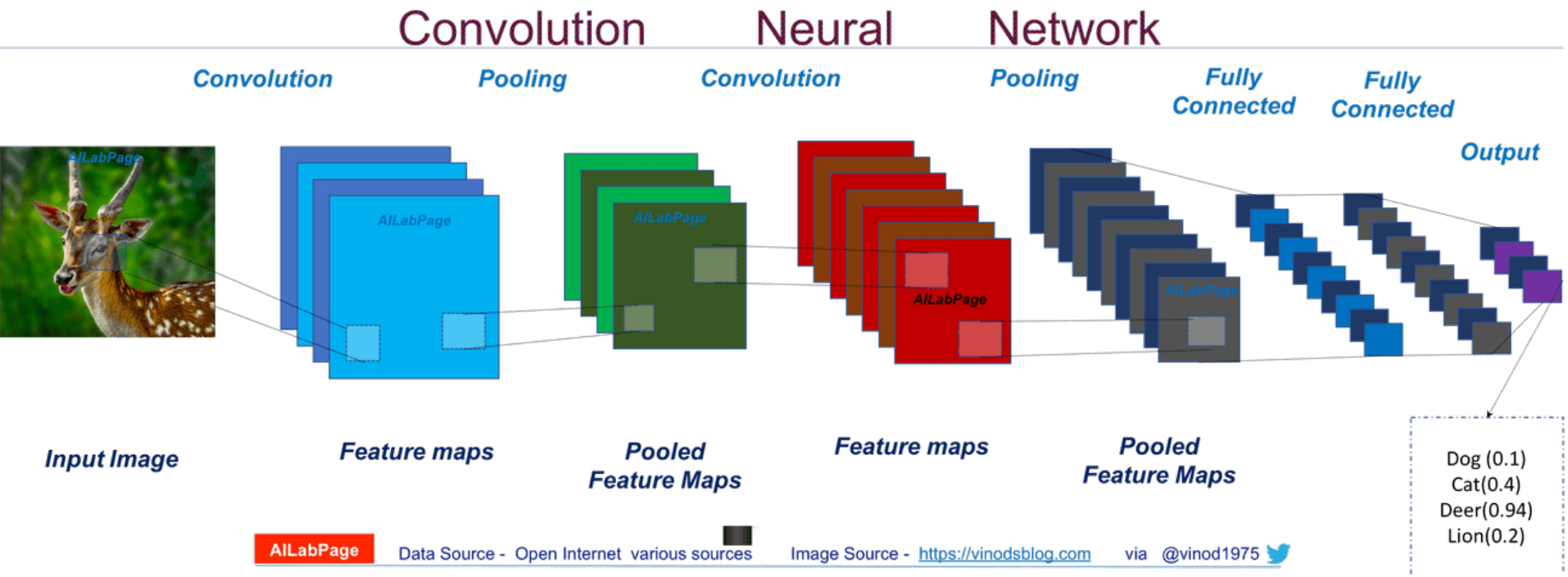






Convolutional Neural Networks

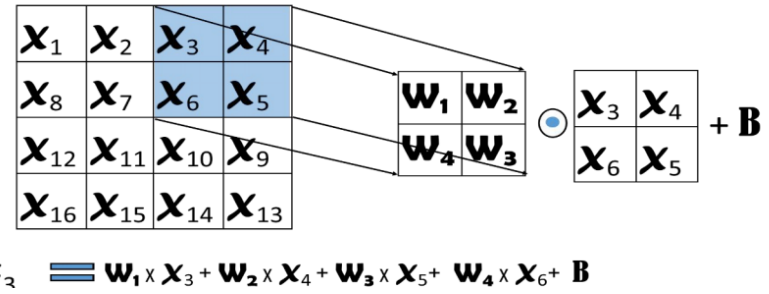
Conceptually: convolve a filter (matrix) across input image to create multiple, transformed representations of the image and use those as input features to a standard NN

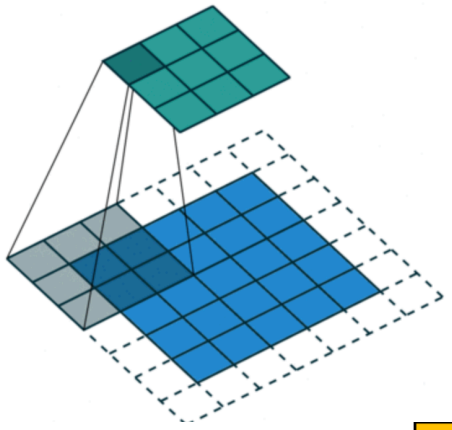


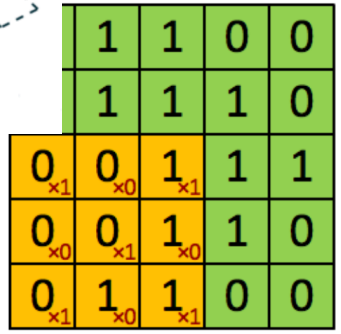
Convolutions

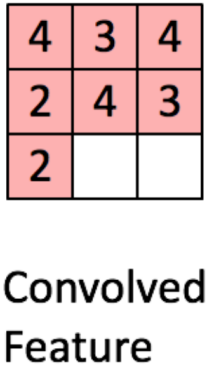
Convolutions apply filter matrices (receptive fields) as a sliding window to input image

- "Pixel" in new image is the dot product of the filter with a portion of the input image
  - The sliding behavior/'shared weights' introduce translation invariance
- Convolutions introduce new hyperparameters:
  - Stride: how many pixels a filter moves at each step
  - Padding: adding 0s to the border of an image so new representation has the same dimensionality





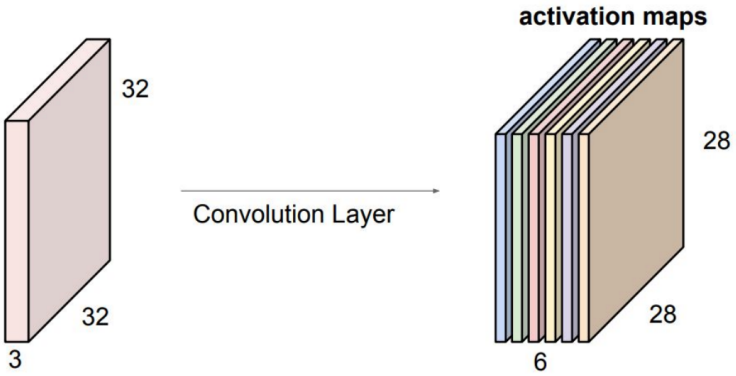


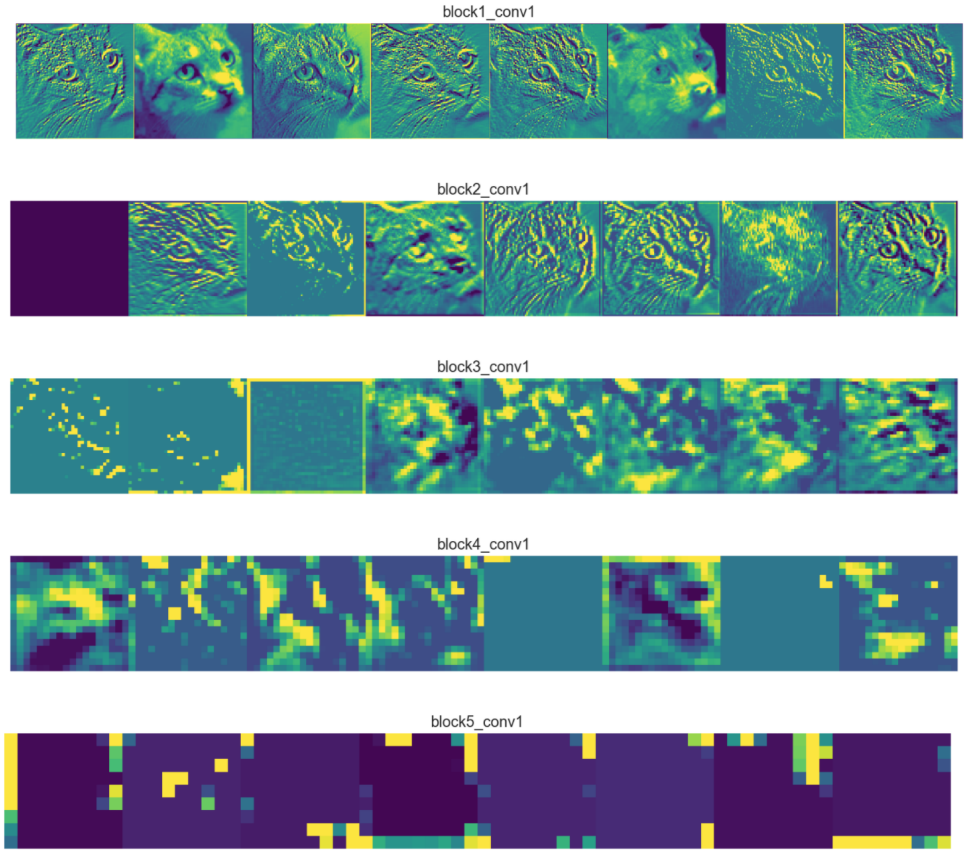


Types of Filters

Each learned filter tries to capture a different image feature (and location relative to other features)

- Typically each layer has multiple convolutions
  - Learn multiple features/orientations
- Each layer becomes more abstracted
  - Like the hierarchical visual cortex!





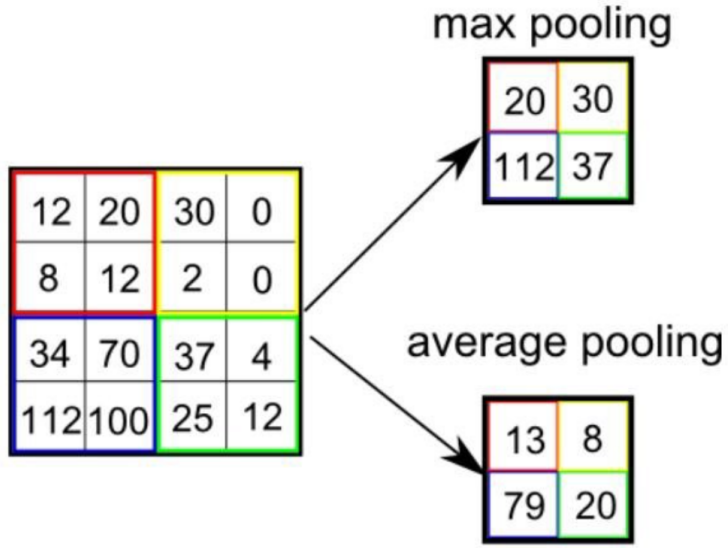
Coding Interlude!

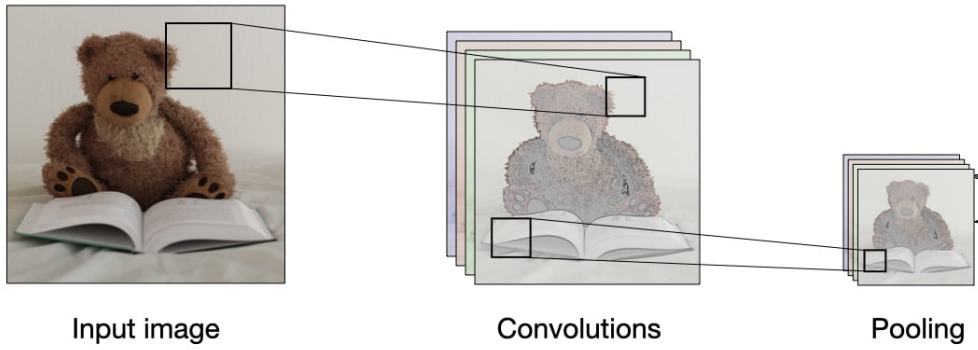
We'll walk through an intuitive tutorial demonstrating how different filters work

Pooling Layers

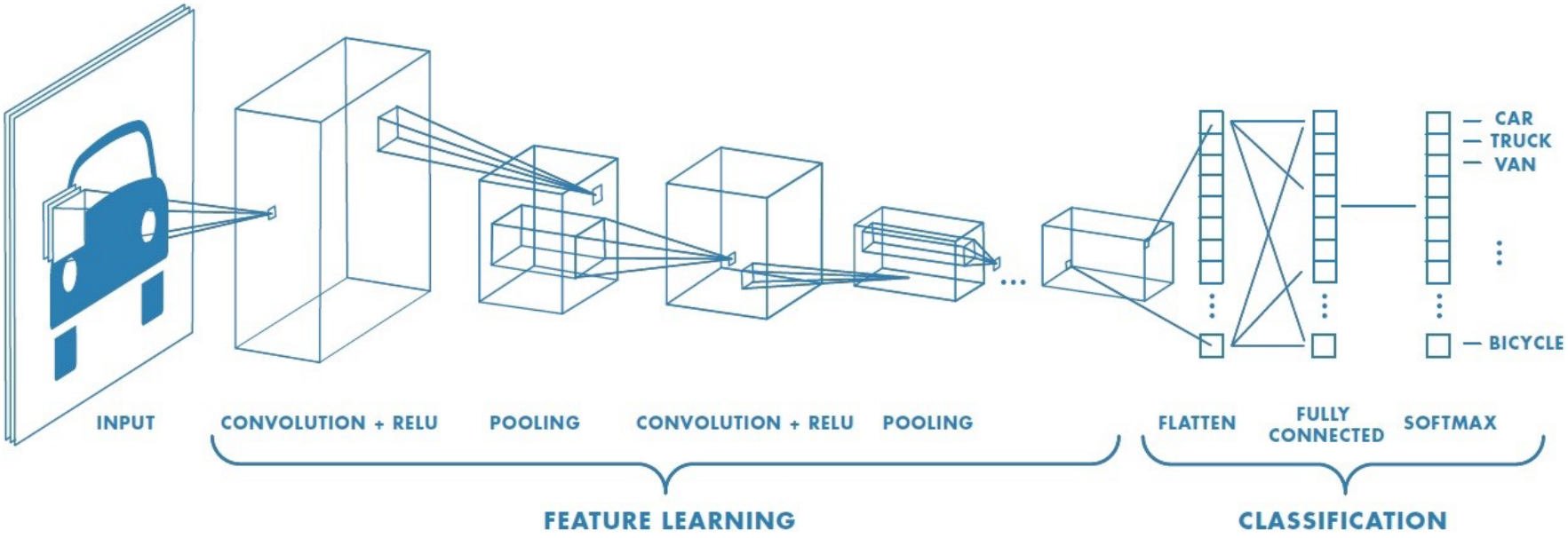
Pooling layers down-sample the new image representations

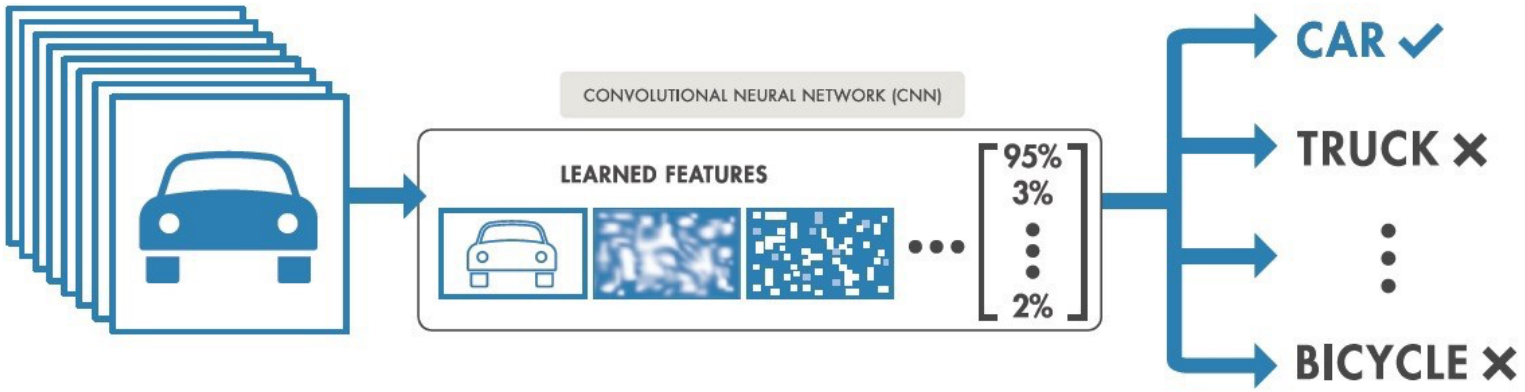
- Typically placed after each convolutional layer
  - Applied as sliding window
- Reduce the size of the convolved features
  - Lower computational load of training and inference
  - Avoid overfitting
  - Enhance translation/rotation invariance
- Two main types:
  - Max returns maximum value from window
  - Average returns average of all values in window





Putting It All Together



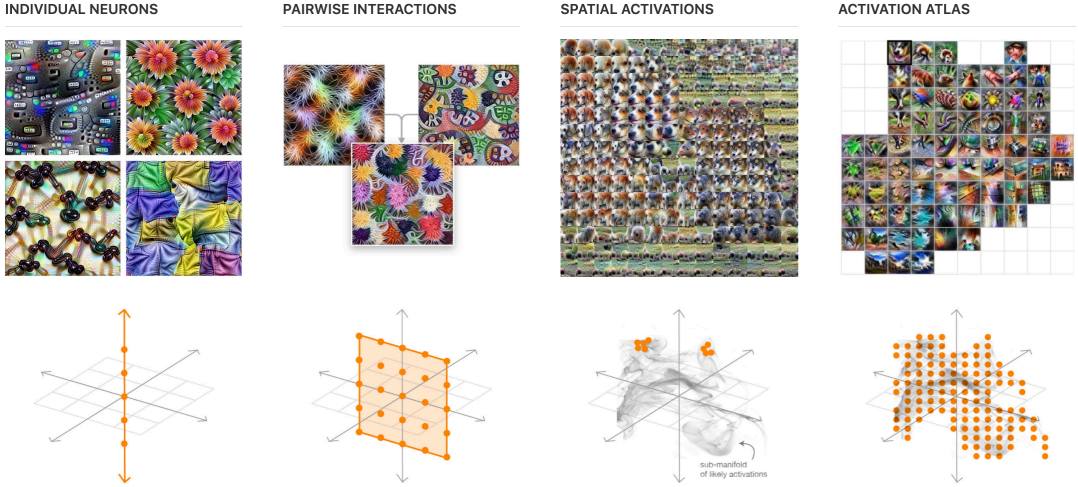


What Is A CNN Learning?

Computer vision is a popular area for interpretability research

Exploring Neural Networks with Activation Atlases

By using feature inversion to visualize millions of activations from an image classification network, we create an explorable *activation atlas* of features the network has learned which can reveal how the network typically represents some concepts.

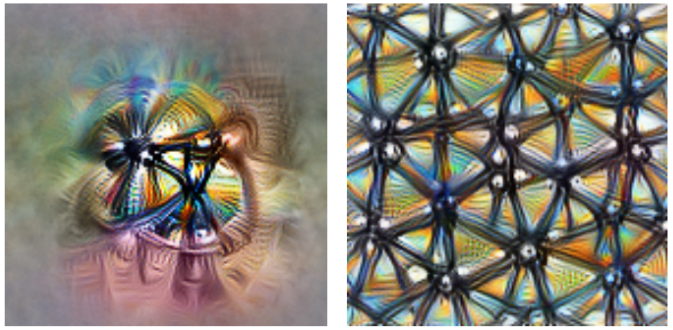


Visualizing individual neurons make hidden layers somewhat meaningful, but misses interactions between neurons — it only shows us one-dimensional, orthogonal probes of the high-dimensional activation space.

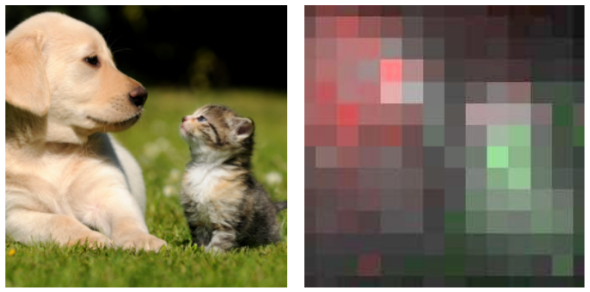
Pairwise interactions reveal interaction effects, but they only show two-dimensional slices of a space that has hundreds of dimensions, and many of the combinations are not realistic.

Spatial activations show us important combinations of many neurons by sampling the sub-manifold of likely activations, but they are limited to those that occur in the given example image.

Activation atlases give us a bigger picture overview by sampling more of the manifold of likely activations.



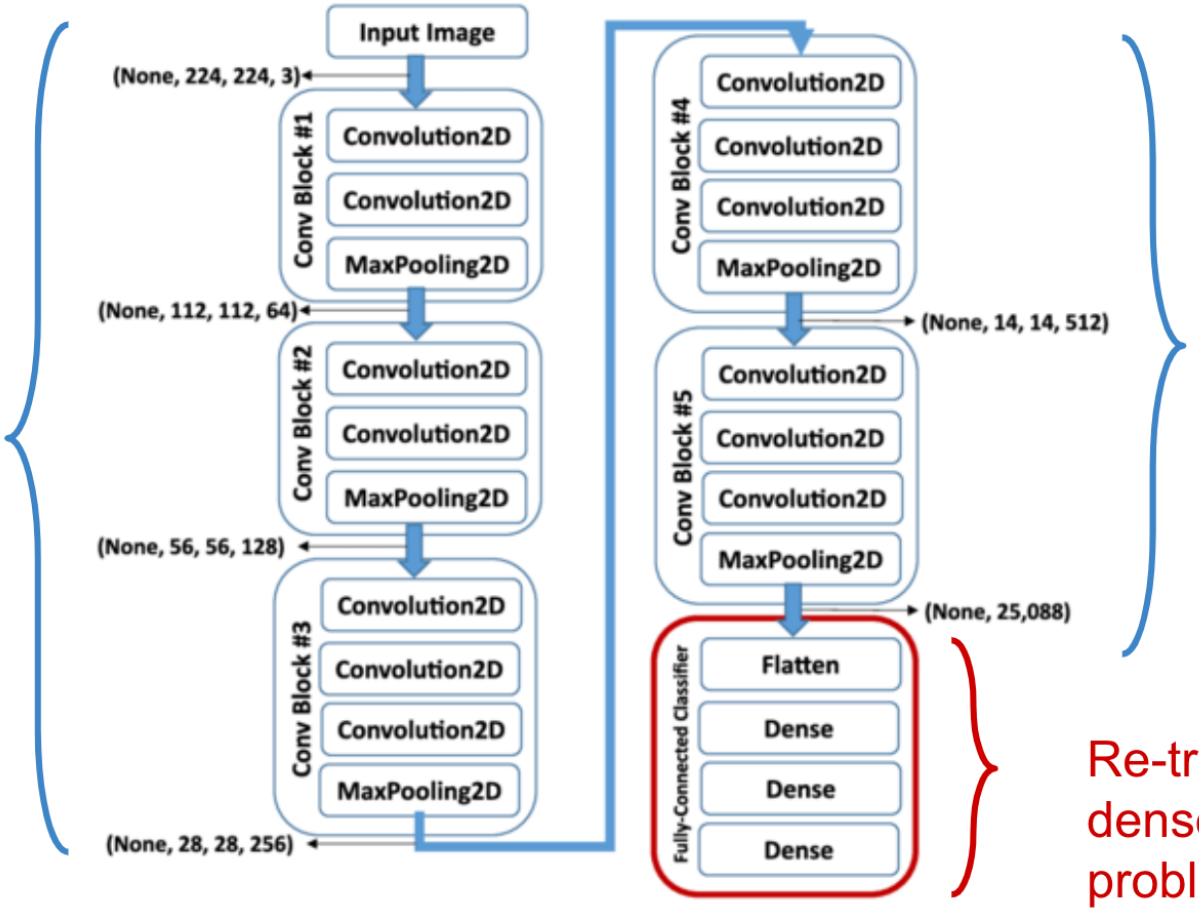
Feature visualization answers questions about what a network — or parts of a network — are looking for by generating examples.



Attribution [1] studies what part of an example is responsible for the network activating a particular way.

Check out some great explanations on Distill!

Transfer Learning



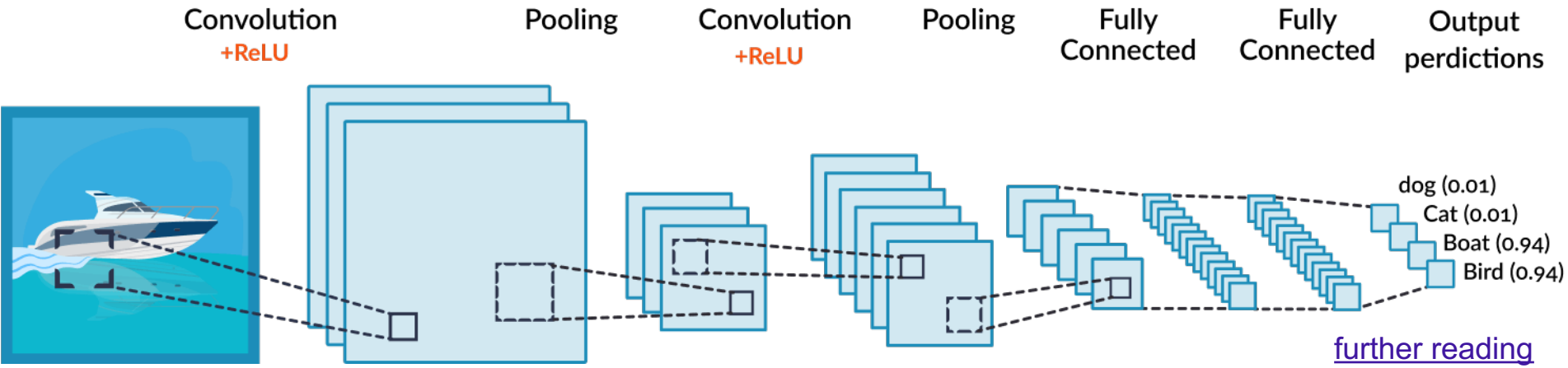
Use pre-trained conv layers (feature extractors). You can also fine tune them on your data

Re-train one or more of the dense layers on your problem

CNN Model Card

Create multiple transformed (convolved) versions of input data

- Convolutions are small, learned matrices that collect local information
- Transformed inputs are typically used in a classification or regression



further reading

- Important parameters:
  - Size and number of convolutions, activations, network structure, rates…
- Data type:
  - fixed size matrices (often used with images)
- Pros:
  - learns locational features, hierarchical features, works with high dimensional data
- Cons:
  - complex architecture, longer training, local minima

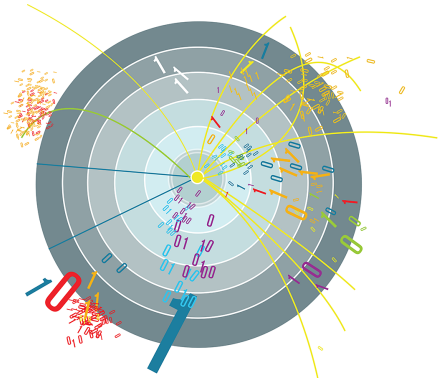
Break!

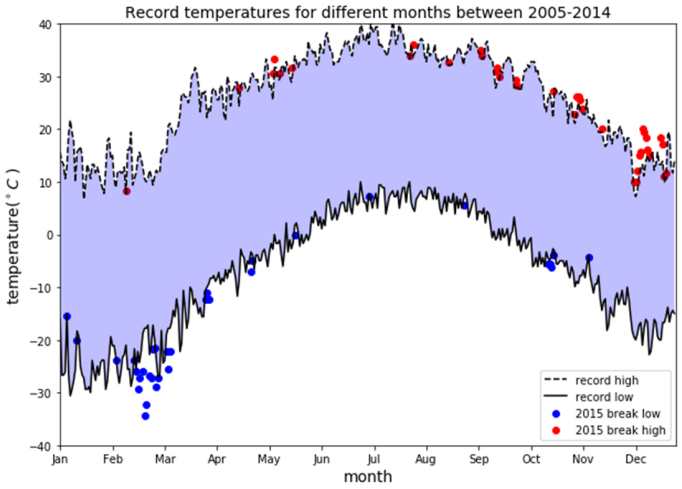
When we return we'll discuss text data and Recurrent NNs

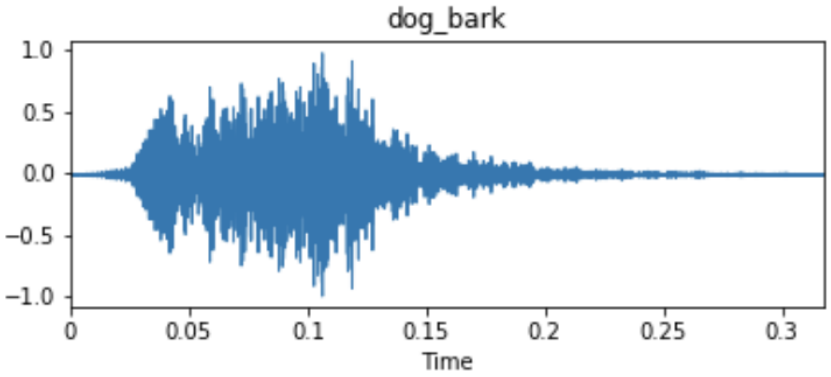
Variable Data

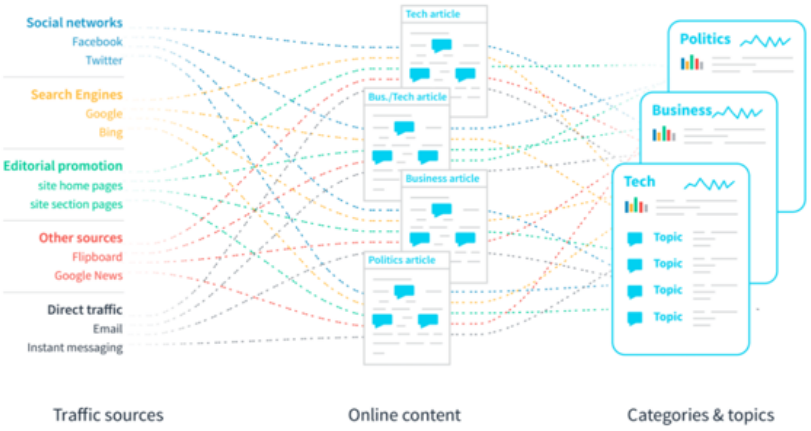
How would the models we've seen so far deal with variable length or dependent data?

- Time based data
  - Historical weather patterns to predict hurricanes
- Text based data
  - Language interpretation or generation
- Audio data
  - Classifying songs
- Data with variable length
  - Particles in a collision





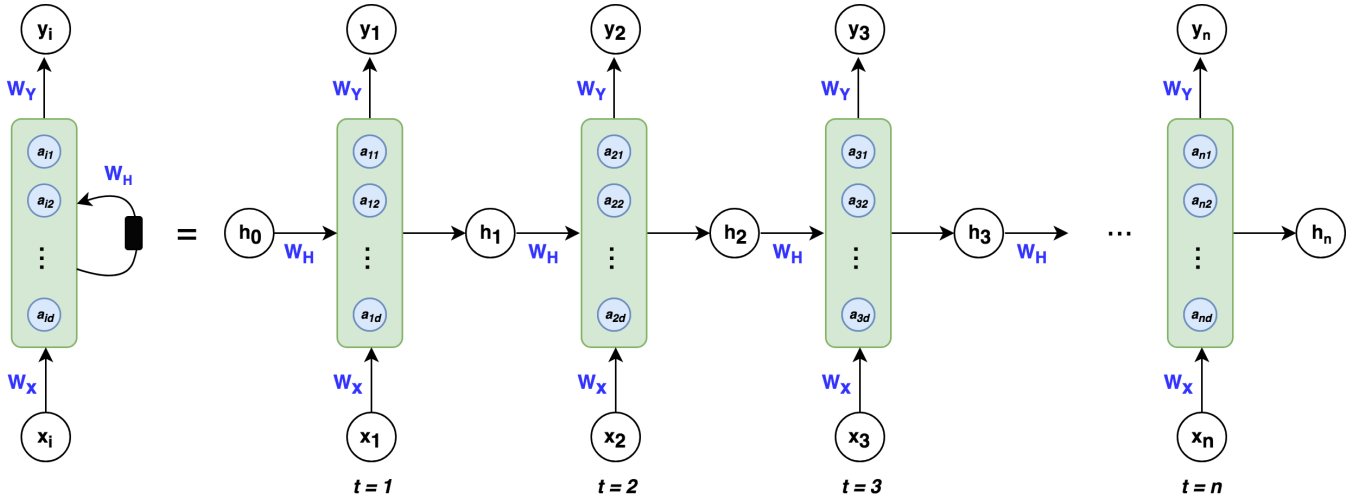




Recurrent Neural Networks

Network nodes have 'memory' that use previous example to process next example

- Unlike standard NNs, don't assume each data point is independent
- Output at current time step/position becomes part of input at next time step/position
- Instead of layers with multiple weights, one neuron is repeated with shared weights
  - Reduces learned parameters
- Key component is 'hidden state' that retains information about sequence



RNN Math

- **Formula for calculating current state:**

$$h_t = f(h_{t-1}, x_t)$$

h_t -> current state
h_{t-1} -> previous state
x_t -> input state

Formula for applying Activation function(tanh):

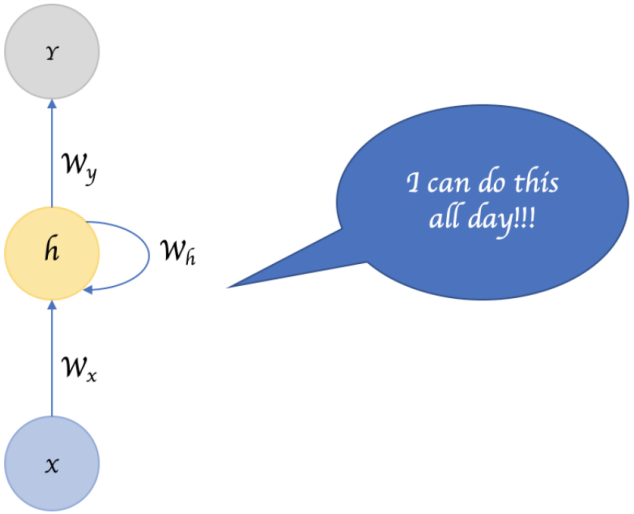
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

W_{hh} -> weight at recurrent neuron
W_{xh} -> weight at input neuron

Formula for calculating output:

$$y_t = W_{hy}h_t$$

Y_t -> output
W_{hy} -> weight at output layer



- h is also called s, both represent hidden state
- Wh, Wx, Wy are the learned parameters

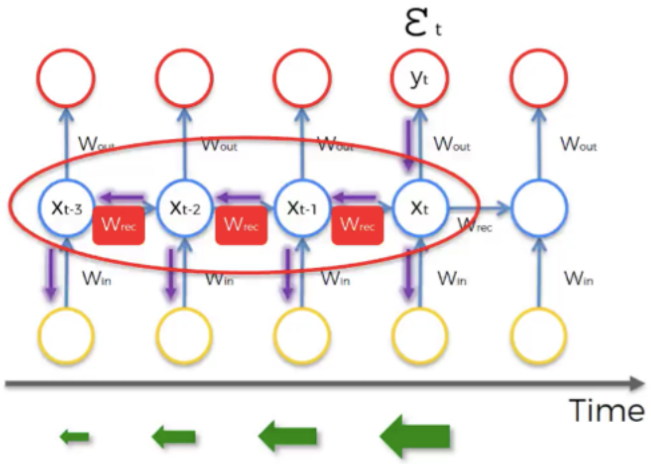
Training RNNs

RNNs are also trained with gradient descent and backpropagation

- Key difference is that back propagation is done at each time point
- Can have outputs for every time step (daily weather prediction) or only for the entire sequence (audio classification)
- Major issue is vanishing or exploding gradients
  - If one timestep has large or small error, it will propagate through entire sequence

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\Big|_{(t)}$$



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

W_{rec} ~ small ⇨ Vanishing

W_{rec} ~ large ⇨ Exploding

Formula Source: Razvan Pascanu et al. (2013)

LSTMS

Long Term Short Term Memory Units address vanishing gradient problem by allowing manipulation of hidden state

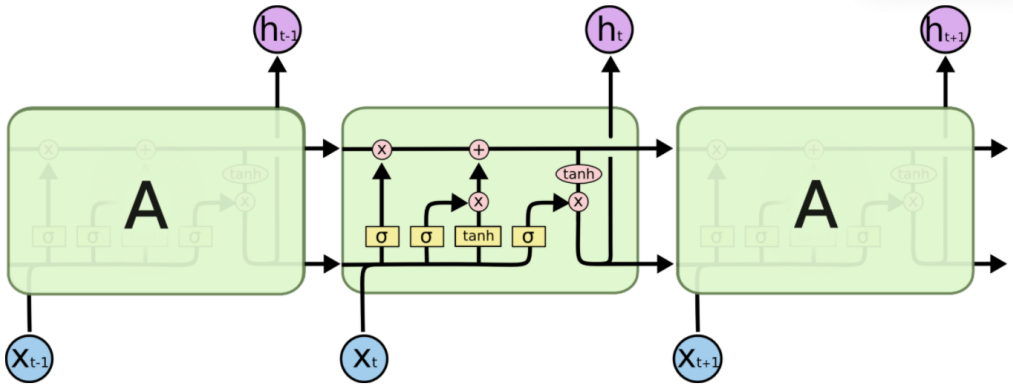
- LSTMs have 4 gates (sigmoids) that determine the flow of data
  - 'Forget gate' looks at previous hidden state and current input to determine which hidden state information to preserve
  - 'Recurrence gate' looks at previous hidden state and current input to decide which information from input to pass on
  - 'Input gate' looks at previous hidden state and current input to decide which parts of hidden state to update
  - 'Output gate' combines information to update the hidden state
- All gates have individual learned weights

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



RNNs and Text

today is your day.
you're off to great places!
you're off and away!

you have brains in your head.
you have feet in your shoes.

you can steer yourself
any direction you choose.
you're on your own. and you know what you know.
and you are the guy who'll decide where to go.

you'll look up and down streets. look 'em over with care.
about some you will say, "i don't choose to go there."
with your head full of brains and your shoes full of feet,
you're too smart to go down any not-so-good street.

and you may not find any
you'll want to go down.
in that case, of course,
you'll head straight out of town.

it's opener there
in the wide open air.

you can learn about trees...
and bees...
and knees.
and knees on trees!
and bees on threes!

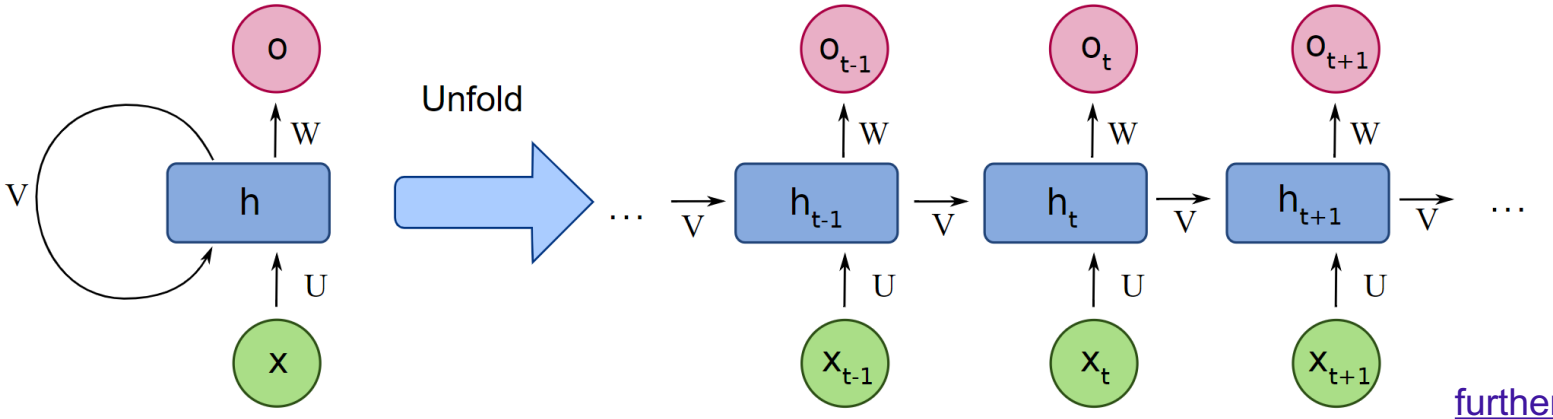
you can read about anchors.
and all about ants.
you can go by cow.
mow we have a mot of inke.
and i hope that they may.
good long, you can learn about.

Overview walk through of generating text with an RNN trained on Alice in Wonderland and Dr. Seuss

RNN Model Card

Output of previous timestep is used to process next timestep

- Network retains ‘memory’ of previous inputs
- Network weights and recurrence weights are updated at each timestep



further reading

- Important parameters:
  - Hidden state size, recurrent unit types, activation functions, directionality, rates…
- Data type:
  - Variable length sequence data (often used with time series, text, video)
- Pros:
  - Allows variable length inputs, maintains sequencing
- Cons:
  - complex architecture, vanishing gradient, limited long-term memory, cannot stack

What topics to discuss Friday?

- Applications of ML in research
- Algorithmic fairness and data bias
- Reinforcement learning
- Benchmark datasets and ML research
- Interpretability
- Policy and regulation of ML
- Advanced computer vision
- Other suggestions?

Coding Time!

Happy to answer any questions!

link to Day 4 notebook

✉ sthais@princeton.edu @basicsciencesav