Computer Vision
grayscale images are matrices

what range of values can each pixel take?

Slides adapted from Mohit Iyyer
color images are tensors

Channels are usually RGB: Red, Green, and Blue
Other color spaces: HSV, HSL, LUV, XYZ, Lab, CMYK, etc
Convolutional Neural Networks
Convolution operator

\[ g(x, y) = \sum_{u} \sum_{v} k(u, v) f(x - u, y - v) \]

(filter, kernel)

Input image * Weights \rightarrow Output image

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>7</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| 11 | 2 | 15 |
| 13 | 8 | 12 |

Slides adapted from Mohit Iyyer
demo:
http://setosa.io/ev/image-kernels/
Convolutional Layer (with 4 filters)

Input: 1x224x224

weights: 4x1x9x9

Output: 4x224x224

if zero padding, and stride = 1

Slides adapted from Mohit Iyyer
Convolutions are commonly used in deep learning to perform image classification and regression. They are composed of two main types of layers: convolutional layers and pooling layers.

**Convolutional Layers:**
- Slide a set of small filters over the image.

**Pooling Layers:**
- Reduce dimensionality of the representation.

Why reduce dimensionality?

Convolutions can be implemented using tools like TensorFlow or PyTorch.

For more information, visit: https://cs231n.github.io/convolutional-networks/
Alexnet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

the paper that started the deep learning revolution!

Slides adapted from Mohit Iyyer
image classification

Classify an image into 1000 possible classes:
e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee,
red fox, banjo, barbell, hourglass, knot, maze, viaduct, etc.

- cat, tabby cat (0.71)
- Egyptian cat (0.22)
- red fox (0.11)

... train on the ImageNet challenge dataset, ~1.2 million images

Slides adapted from Mohit Iyyer
Alexnet

conv+pool

conv+pool

conv
conv
conv

linear
linear

linear+
softmax

https://www.saagie.com/fr/blog/object-detection-part1

Slides adapted from Mohit Iyyer
What is happening?

[Diagram of a deep neural network with layers labeled: Input Layer, Hidden Layer 1, Hidden Layer 2, Hidden Layer 3, Output Layer. Below the diagram, images of edges, combinations of edges, and object models.]

https://www.saagie.com/fr/blog/object-detection-part1

Slides adapted from Mohit Iyyer
at the end of the day, we generate a fixed size vector from an image and run a classifier over it

\[
CNN(\text{image}) = \text{softmax: predict ‘truck’}
\]
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    x = layers.Conv2D(8, (5, 5), activation='relu', strides=1)(inputs)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(16, (5, 5), activation='relu', strides=1)(x)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Flatten()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)

def make_model(input_shape=image_size+(1,), num_classes=2):
    keras.utils.plot_model(model, show_shapes=True)
New Architecture

- **Input Layer**
  - Input: $[(None, 180, 180, 1)]$
  - Output: $[(None, 180, 180, 1)]$

- **Conv2D (conv2d_9)**
  - Input: $(None, 180, 180, 1)$
  - Output: $(None, 176, 176, 8)$

- **MaxPooling2D (max_pooling2d_2)**
  - Input: $(None, 176, 176, 8)$
  - Output: $(None, 88, 88, 8)$

- **Conv2D (conv2d_10)**
  - Input: $(None, 88, 88, 8)$
  - Output: $(None, 84, 84, 16)$

- **MaxPooling2D (max_pooling2d_3)**
  - Input: $(None, 84, 84, 16)$
  - Output: $(None, 42, 42, 16)$

- **Flatten (flatten_3)**
  - Input: $(None, 42, 42, 16)$
  - Output: $(None, 28224)$

- **Dense (dense_1)**
  - Input: $(None, 28224)$
  - Output: $(None, 1)$
Auto-Encoders
Auto-encoders

Auto-encoders are a class of neural networks that do not require labeled data.

**Supervised NNs**: predict the output given the input.

**Auto-encoders**: predict the input given the input.

**Key idea**: select features by reducing then increasing dimensionality.
Normal NN goes:

Auto-encoder goes:
Auto-Encoders as Dimensionality Reduction

Auto-encoders are a more powerful form of dimensionality reduction than traditional techniques like PCA, because they can learn nonlinear transformations.

www.jeremyjordan.me/autoencoders
Model: "sequential_6"

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d_9 (Conv2D)</td>
<td>(None, 32, 32, 32)</td>
<td>320</td>
</tr>
<tr>
<td>dropout_18 (Dropout)</td>
<td>(None, 32, 32, 32)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_10 (Conv2D)</td>
<td>(None, 16, 16, 64)</td>
<td>18496</td>
</tr>
<tr>
<td>dropout_19 (Dropout)</td>
<td>(None, 16, 16, 64)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_11 (Conv2D)</td>
<td>(None, 8, 8, 128)</td>
<td>73856</td>
</tr>
<tr>
<td>dropout_20 (Dropout)</td>
<td>(None, 8, 8, 128)</td>
<td>0</td>
</tr>
<tr>
<td>flatten_3 (Flatten)</td>
<td>(None, 8192)</td>
<td>0</td>
</tr>
<tr>
<td>dense_6 (Dense)</td>
<td>(None, 128)</td>
<td>1048704</td>
</tr>
</tbody>
</table>

Total params: 1,141,376
Trainable params: 1,141,376
Non-trainable params: 0
## Decoder

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_7 (Dense)</td>
<td>(None, 8192)</td>
<td>1056768</td>
</tr>
<tr>
<td>reshape_3 (Reshape)</td>
<td>(None, 8, 8, 128)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_transpose_12 (Conv2D Transpose)</td>
<td>(None, 16, 16, 128)</td>
<td>147584</td>
</tr>
<tr>
<td>dropout_21 (Dropout)</td>
<td>(None, 16, 16, 128)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_transpose_13 (Conv2D Transpose)</td>
<td>(None, 32, 32, 64)</td>
<td>73792</td>
</tr>
<tr>
<td>dropout_22 (Dropout)</td>
<td>(None, 32, 32, 64)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_transpose_14 (Conv2D Transpose)</td>
<td>(None, 64, 64, 32)</td>
<td>18464</td>
</tr>
<tr>
<td>dropout_23 (Dropout)</td>
<td>(None, 64, 64, 32)</td>
<td>0</td>
</tr>
<tr>
<td>conv2d_transpose_15 (Conv2D Transpose)</td>
<td>(None, 64, 64, 1)</td>
<td>289</td>
</tr>
</tbody>
</table>

Total params: 1,296,897
Trainable params: 1,296,897
Non-trainable params: 0
Input, Output, Difference

Epoch 1

Epoch 10
Using Decoder to Generate

Input noise to the decoder to make it hallucinate a cat:

```python
x = autoencoder.decoder(np.random.randn(1, 128)).numpy()
plt.imshow(x[0, :, :, 0], cmap='gray')
```
Stable Diffusion
Stable Diffusion

3 components:

1. VAE: an auto-encoder to map images to a latent space
2. U-Net: an architecture that learns to denoise images
3. CLIP: a text-encoder to allow multi-modal input
VAE: variational autoencoder

VAE is an encoder/decoder model.

The encoder maps an input image (pixels) to a lower-dimension latent space.

The decoder takes the output of the model and maps it back to an image in pixels.
U-Net model (auto-encoder)

Iteratively Denoising
CLIP: a text encoder for multi-modal input

Objective: given a batch of text and image inputs, predict the correct image-text pairings.

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford*1  Jong Wook Kim*1  Chris Hallacy1  Aditya Ramesh1  Gabriel Goh1  Sandhini Agarwal1
Girish Sastry1  Amanda Askell1  Pamela Mishkin1  Jack Clark1  Gretchen Krueger1  Ilya Sutskever1

Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image can be used to train powerful visual models. Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of “text-to-text” as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset engineering.
CLIP: a text encoder for multi-modal input

1. Contrastive pre-training

2. Create dataset classifier from label text

3. Use for zero-shot prediction
Stable Diffusion: putting the pieces together

repeat 50 times

User Prompt
"An astronaut riding a horse"

Text conditioned U-Net
64x64 conditioned latents

TextEncoder
77x788 text embeddings

Frozen CLIP Text Encoder

Latent Seed
Gaussian noise ∼N(0,1)

64x64 latents

Variational Autoencoder Decoder
512x512 output image

Scheduler algorithm: "reconstruct"

repeat N scheduler steps