BST 261: Data Science II
Lecture 5

Introduction to Convolutional Neural Networks (CNNs)

Heather Mattie
Harvard T.H. Chan School of Public Health
Spring 2 2020

Recipe of the Day!

[Triple Berry Pie]

Paper Presentations

CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

Pranav Rajpurkar [*1] Jeremy Irvin [*1] Kaylie Zhu [1] Brandon Yang [1] Hershel Mehta [1]
Tony Duan [1] Daisy Ding [1] Aarti Bagul [1] Robyn L. Ball [2] Curtis Langlotz [3] Katie Shpanskaya [3]
Matthew P. Lungren [3] Andrew Y. Ng [1]

Tim Genovese, MPH'20

Background

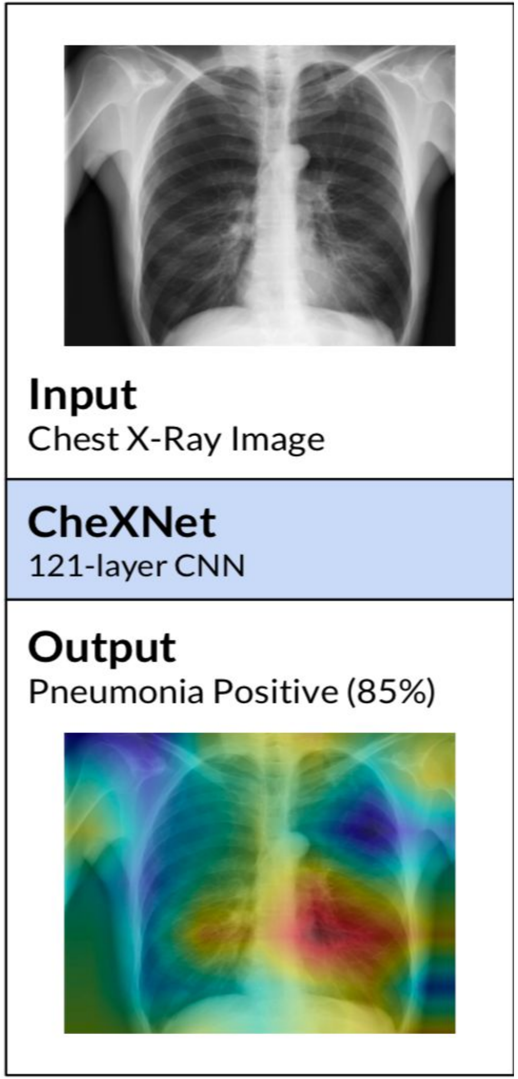
CXR is one of the most common medical tests

First-line test for many chest diseases, including pneumonia

More than 1 million hospitalizations for pneumonia in the US per year

CheXNet Model

- 121-Layer Convolutional Neural Network (CNN)
- Dense Connections
- Input: Frontal Chest X-ray Image (224x224)
- Output: Probability of Pneumonia
- Activation = Sigmoid, Loss = Binary Cross-Entropy
- Adam Optimizer



Dataset

- ChestX-ray14 Dataset (NIH)
- 112,120 frontal-view CXR
- 30,805 unique patients
- Up to 14 different pathologies annotated from radiology reports

Pathology
Atelectasis
Cardiomegaly
Effusion
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax
Consolidation
Edema
Emphysema
Fibrosis
Pleural Thickening
Hernia

Training

- Weights were initialized by pretraining on ImageNet database
- Dataset randomly split:
  - Training (28744 patients, 98637 images)
  - Validation (1672 patients, 6351 images)
  - Test (389 patients, 420 images)
  - No patient overlap between the sets.
- Augmented training data with random horizontal flipping
- Batch normalization of inputs

Comparison

- 4 Stanford radiologists (5, 7, 25, 28 years experience)
  - Performed binary classification on 420 images
  - No clinical or additional information
- Outcome: F1 Statistic

$$F_\beta = \frac{(1+\beta^2)\cdot \text{true positive}}{(1+\beta^2)\cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}}$$

Results

- CheXNet outperformed the radiologists' average for pneumonia

	F1 Score (95% CI)
Radiologist 1	0.383 (0.309, 0.453)
Radiologist 2	0.356 (0.282, 0.428)
Radiologist 3	0.365 (0.291, 0.435)
Radiologist 4	0.442 (0.390, 0.492)
Radiologist Avg.	0.387 (0.330, 0.442)
CheXNet	0.435 (0.387, 0.481)

- Difference in F1 scores: 0.051 (

Extension: 14 Pathologies

- Same input and model, output layer changed to predict 14 pathologies

Pathology	Wang et al. (2017)	Yao et al. (2017)	CheXNet (ours)
Atelectasis	0.716	0.772	**0.8094**
Cardiomegaly	0.807	0.904	**0.9248**
Effusion	0.784	0.859	**0.8638**
Infiltration	0.609	0.695	**0.7345**
Mass	0.706	0.792	**0.8676**
Nodule	0.671	0.717	**0.7802**
Pneumonia	0.633	0.713	**0.7680**
Pneumothorax	0.806	0.841	**0.8887**
Consolidation	0.708	0.788	**0.7901**
Edema	0.835	0.882	**0.8878**
Emphysema	0.815	0.829	**0.9371**
Fibrosis	0.769	0.767	**0.8047**
Pleural Thickening	0.708	0.765	**0.8062**
Hernia	0.767	0.914	**0.9164**

- Outcome is AURO(

Limitations

- Radiologists were missing important info:
    - Lateral View Chest X-Ray
    - Demographic (age, sex)
    - Clinical (fever, chest pain, duration)
- Misclassification of the ground truth (taken from radiology reports)

INDICATION: ___ year old woman with ?pleural effusion // ?pleural effusion

TECHNIQUE: Chest PA and lateral

COMPARISON: ___

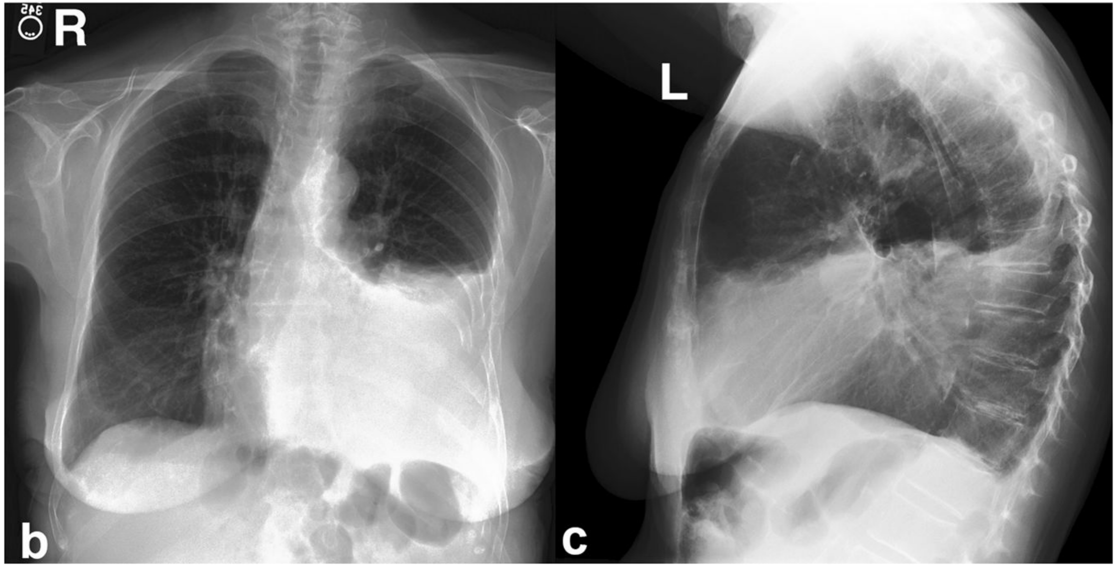
FINDINGS:

Cardiac size cannot be evaluated. Large left pleural effusion is new. Small right effusion is new. The upper lungs are clear. Right lower lobe opacities are better seen in prior CT. There is no pneumothorax. There are mild degenerative changes in the thoracic spine

IMPRESSION:

Large left pleural effusion

a



Johnson et al, 2019

References

- Johnson, A.E.W., Pollard, T.J., Berkowitz, S.J. *et al.* MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports. *Sci Data* **6,** 317 (2019). https://doi.org/10.1038/s41597-019-0322-0

- Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, DuanT, Ding D, Bagul A, Ball RL, Langlotz C, Shpanskaya K, Lungren MP, Ng AY. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. https://arxiv.org/abs/1711.05225

SHAN (CHLOE) HE

BST 261

DERMATOLOGIST-LEVEL CLASSIFICATION OF SKIN-CANCER WITH DEEP NEURAL NETWORKS

BACKGROUND

One in five Americans will be diagnosed with a skin malignancy in their lifetime

Early detection is critical

Primarily diagnosed visually

Challenging – Variabilities in skin lesions

PREVIOUSLY …

Insufficient Sample Size

Focus extensively on highly **standardized images**, thus poor classification on photographic images

Require extensive preprocessing

THIS PAPER …

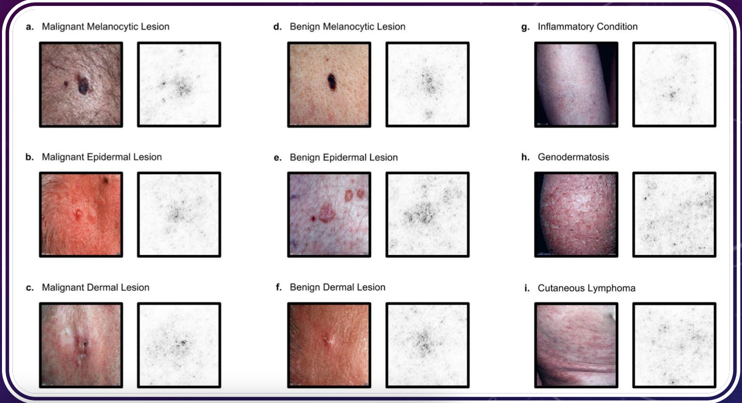
Larger sample size: 129450 images, 2032 different diseases

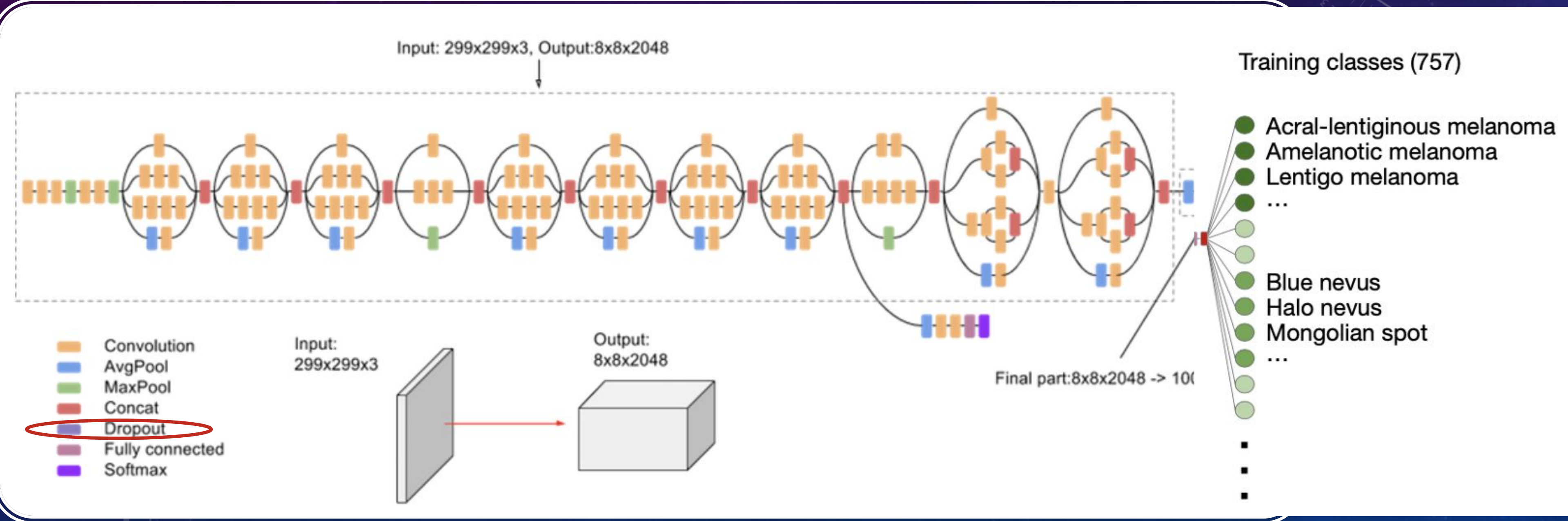
1.41 million pre-training and training images make classification robust to **photographic variability**

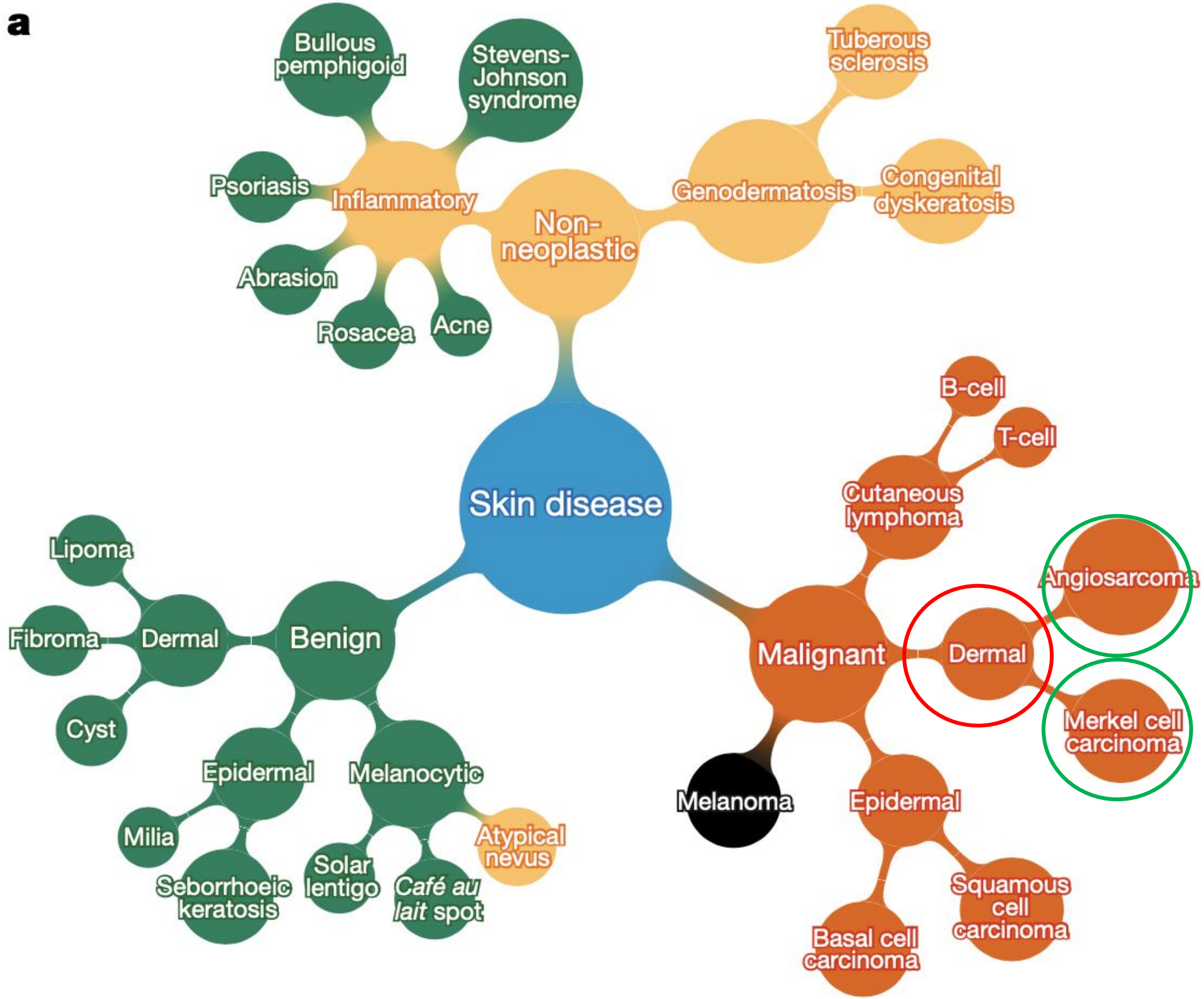
No handcrafted features

METHOD

- TRANSFER LEARNING: Google's Inception v3 CNN Retrain the final classification layer
- Fine-tune the parameters across all layers using backpropagation

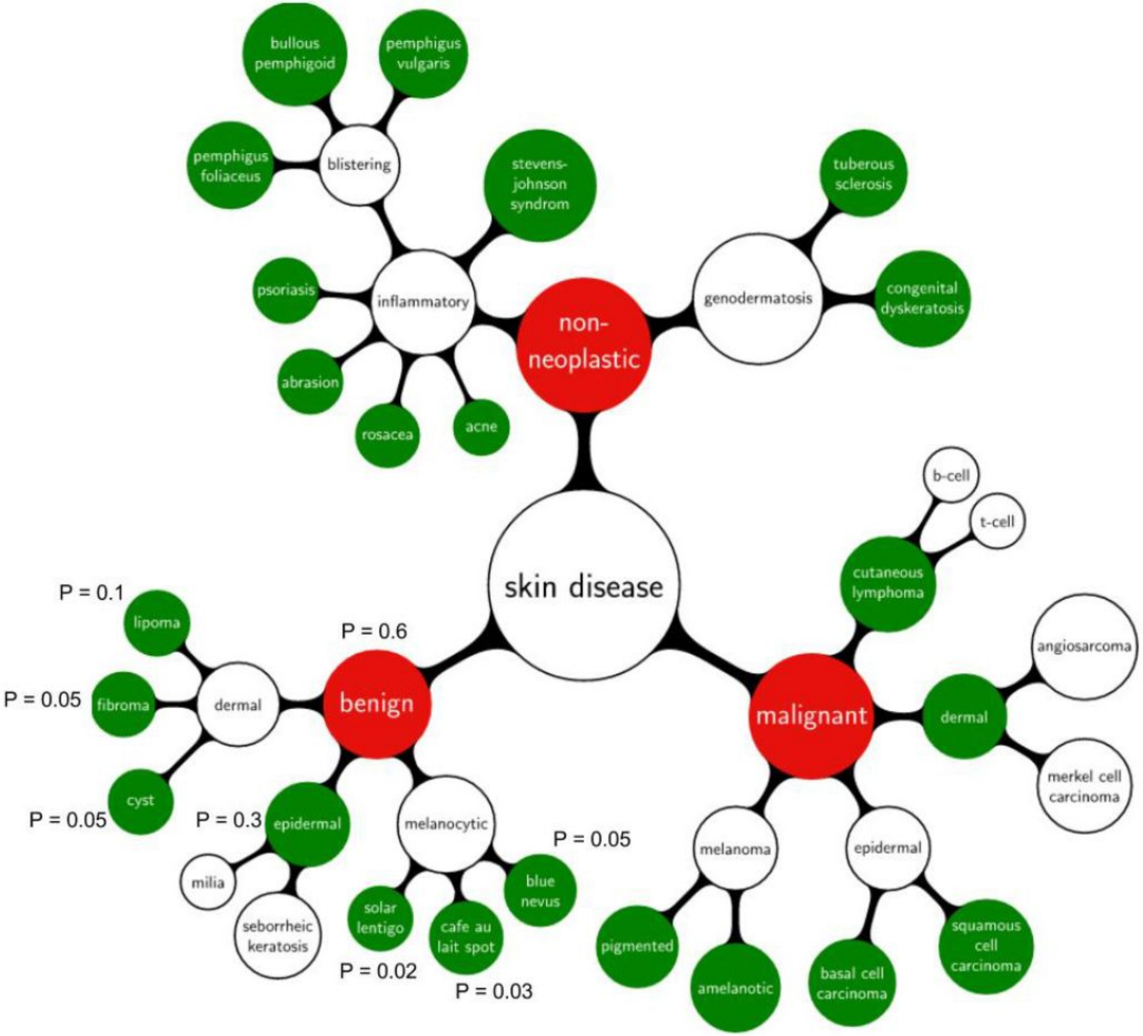




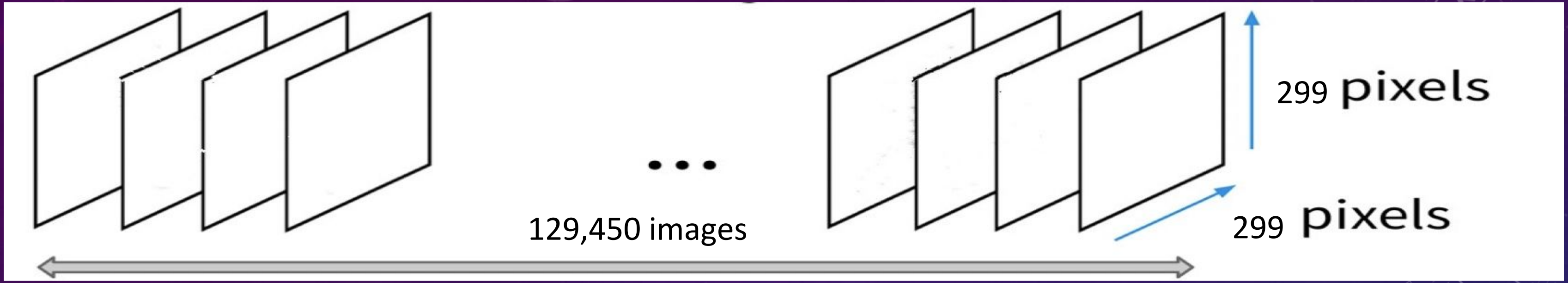


Method Intuition

- P (each leaf node)
- P (Coarser Level) = sum P (child notes)



- Specify: max class size --- maximum number of data points in a class
- The algorithm begins with the top node and recursively descends the taxonomy, turning nodes into training classes if the amount of data contained in that node does not exceed a specified threshold
- Avoids having training classes that are overly fine grained and that do not have sufficient data to be learned properly



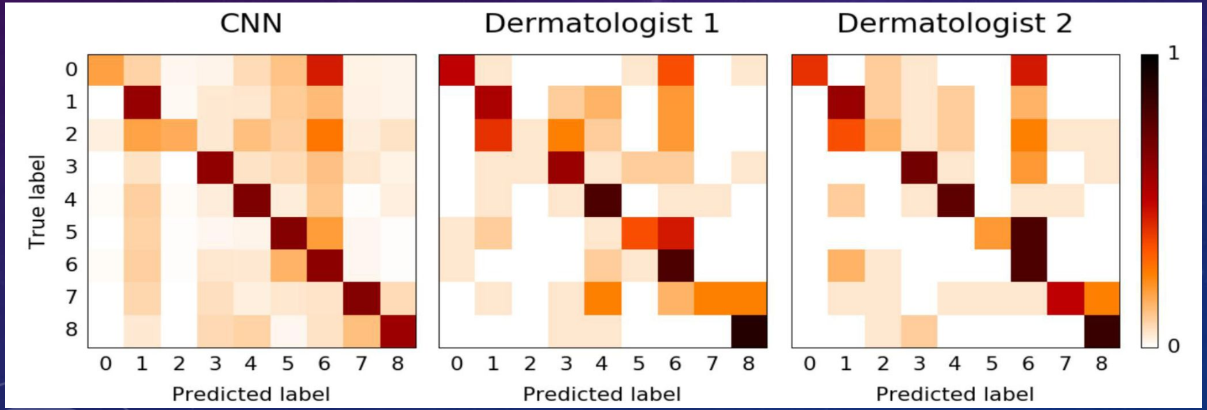
127,508 training and validation images
* labelled by dermatologists

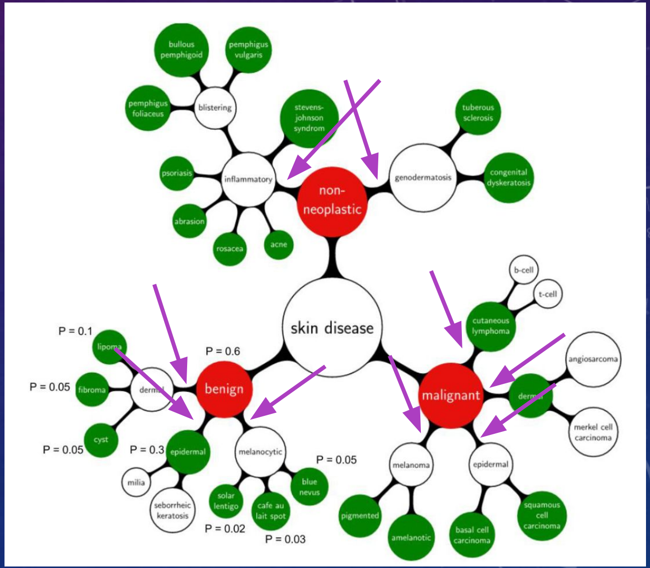
1,942 test images
*biopsy-labelled

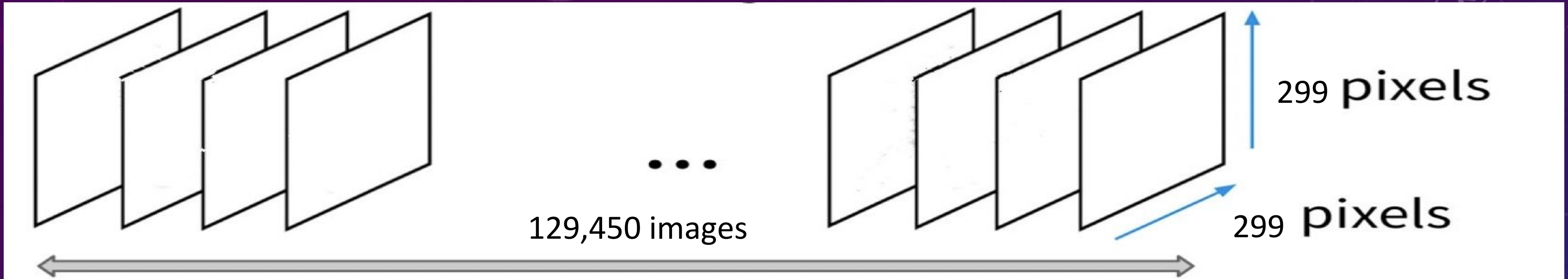
Validation Set
(9-fold)

PERFORMANCE ON VALIDATION SET

Accuracy	3 classes (3 red root nodes)	9 classes (children of 3 red root nodes)
CNN	72%	55%
Two Dermatologist	65%	54%



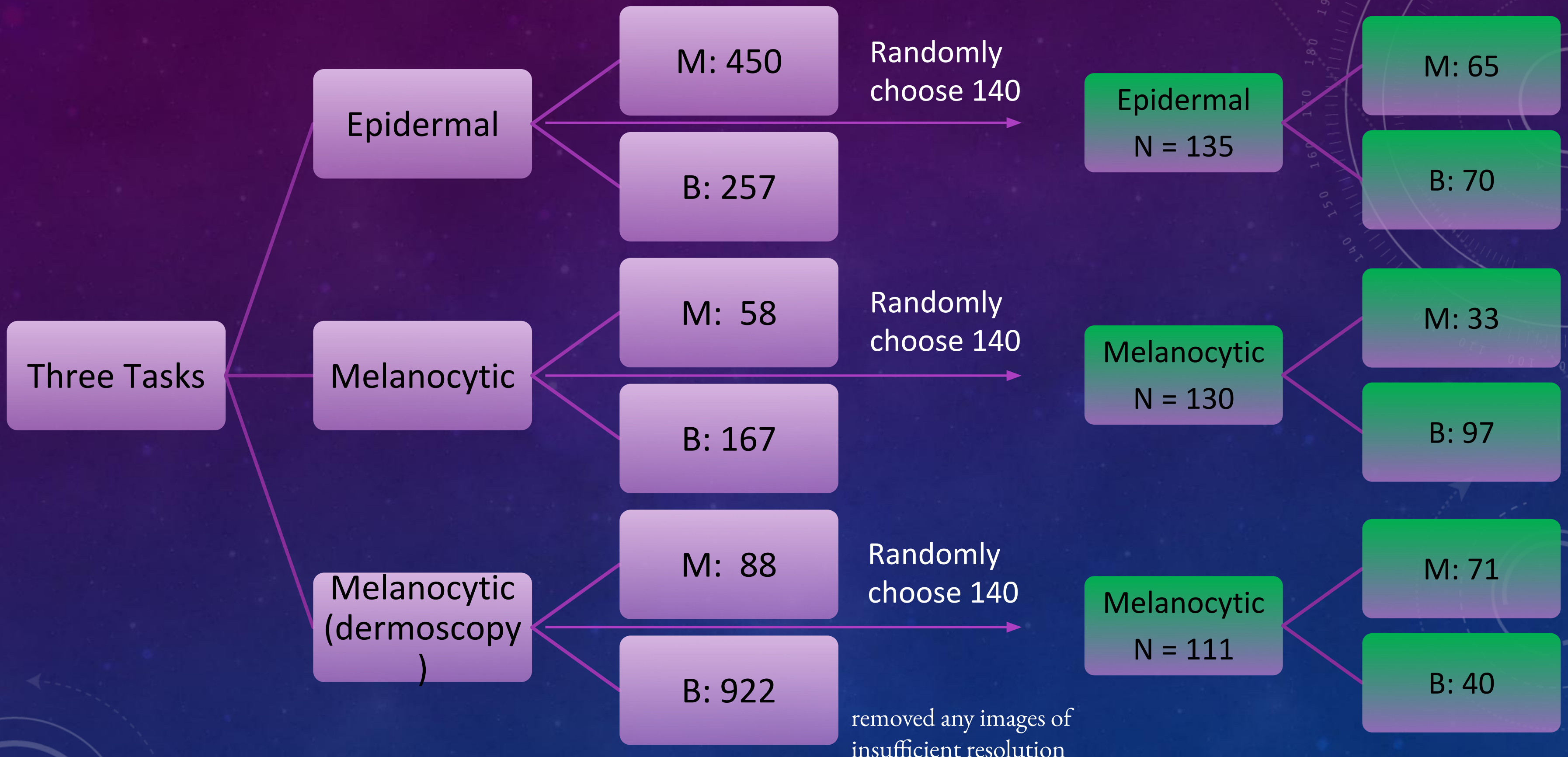




127,508 training and validation images
* labelled by dermatologists

1,942 test images
*biopsy-labelled

Validation Set
(9-fold)

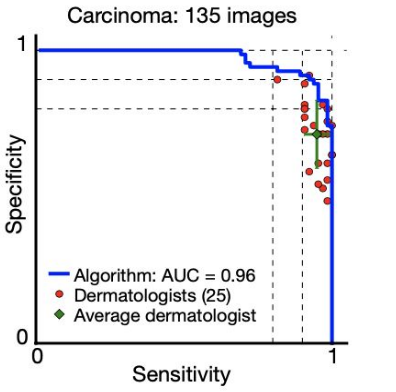
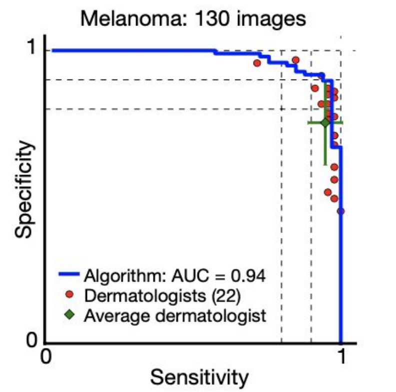
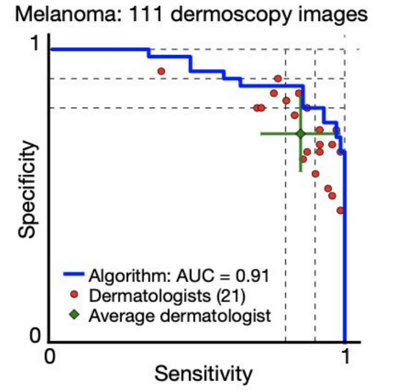


removed any images of insufficient resolution (dermatologists require high resolution)

PERFORMANCES ON TEST SET

Using dermoscopy images

*biopsy-proven

<table>
<tr><th>AUC</th><th>Benign vs Malignant epidermal lesions</th><th>Benign vs Malignant Melanocytic lesions</th><th>Benign vs Malignant Melanocytic lesions</th></tr>
<tr><td>Smaller Test Data</td><td>
</td><td>
</td><td>
</td></tr>
<tr><td>Full Test Data</td><td>96%</td><td>96%</td><td>94%</td></tr>
</table>

TAKEAWAYS

- CNN outperforms dermatologist
- This algorithm is reliable on a larger dataset (Generalizability)
- Classification Algorithm

  \+

  Contextual Expertise

Convolutional Neural Networks (CNNs)

CNNs

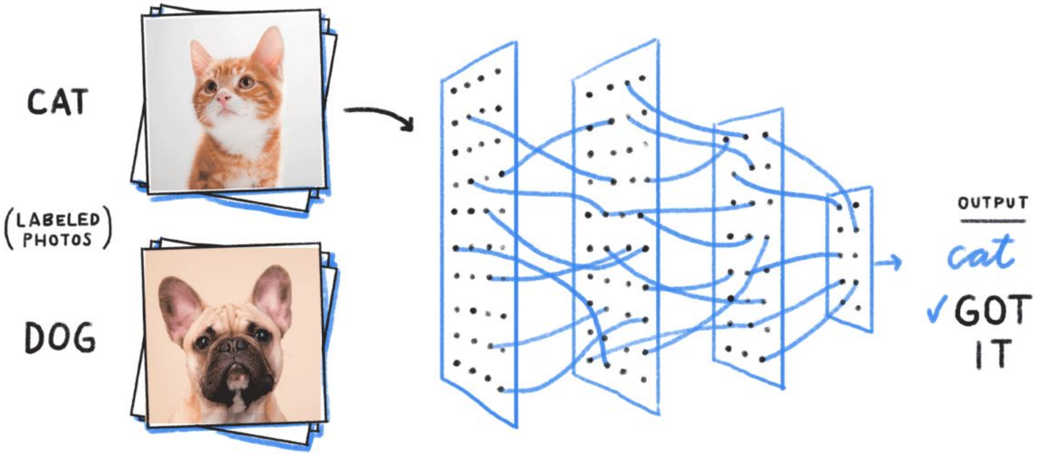
Convolutional neural networks = CNNs = convnets = (many times) computer vision

CNNs are at the heart of deep learning, emerging in recent years as the most prominent strain of neural networks in research. They have revolutionized computer vision, achieving state-of-the-art results in many fundamental tasks, as well as making strong progress in natural language processing, reinforcement learning, and many other areas. CNNs have been widely deployed by tech companies for many of the new services and features we see today. They have numerous and diverse applications, including:

- Detecting and labeling objects, locations, and people in images
- Converting speech into text and synthesizing audio of natural sounds
- Describing images and videos with natural language
- Tracking roads and navigating around obstacles in autonomous vehicles
- Analyzing video game screens to guide autonomous agents playing them

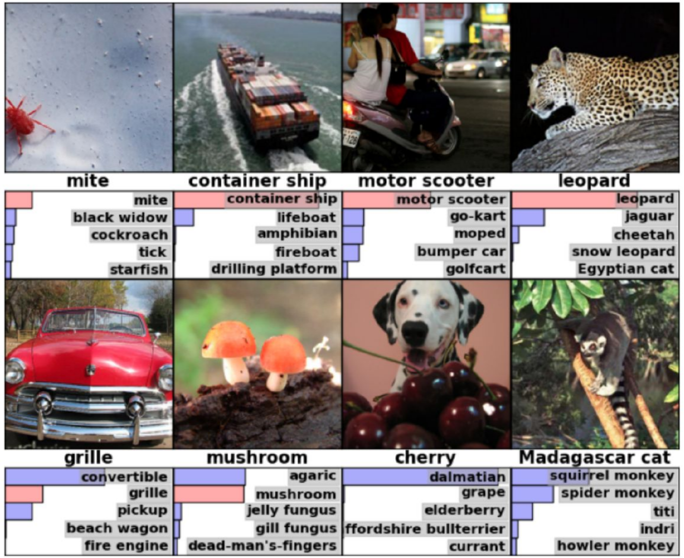
CNN Applications

Image classification



CNN Applications

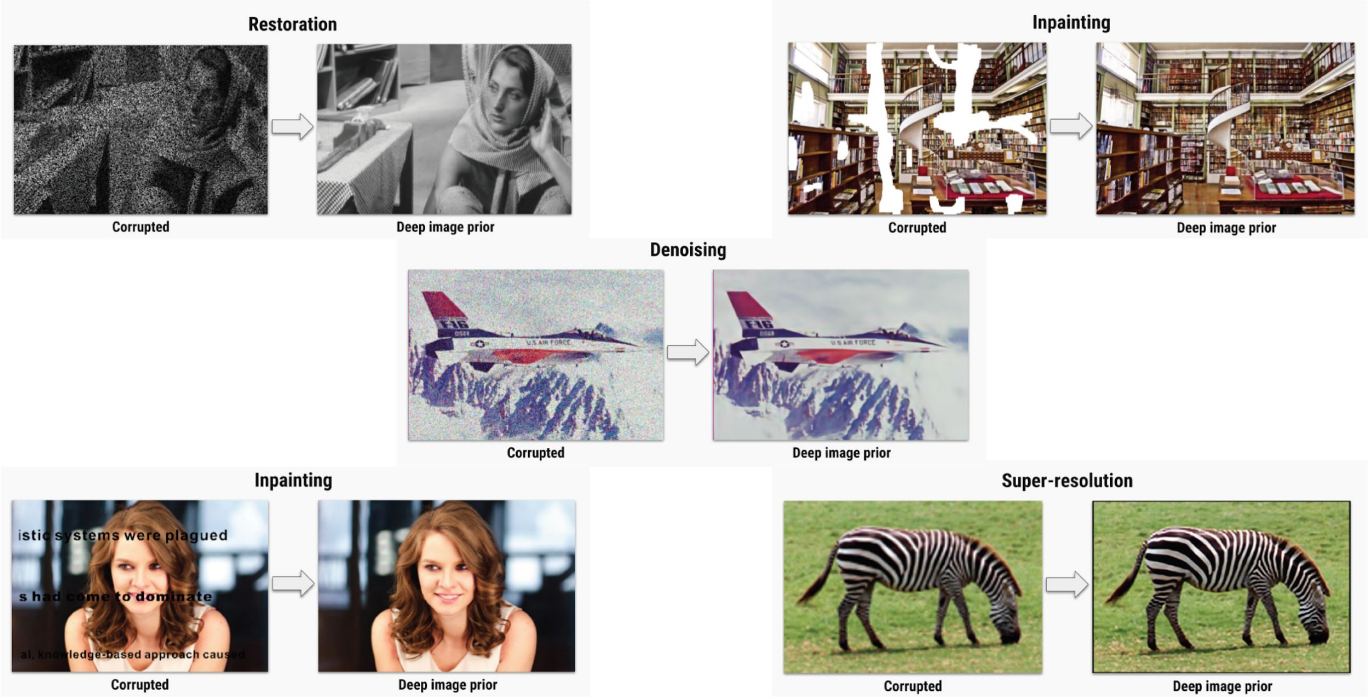
Computer vision



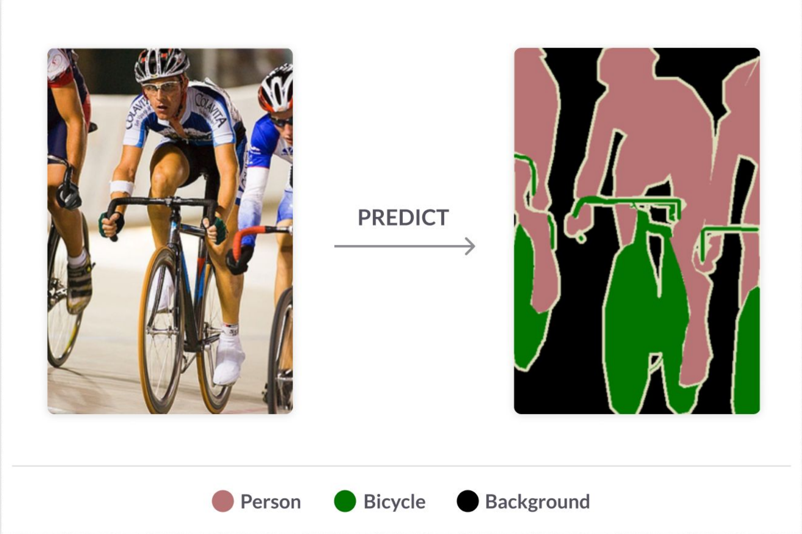
Top: 4 correctly classified examples. Bottom: 4 incorrectly classified examples. Each example has an image, followed by its label, followed by the top 5 guesses with probabilities. From Krizehvsky *et al.* (2012).

CNN Applications

Restoration, inpainting, denoising and super-resolution



CNN Applications



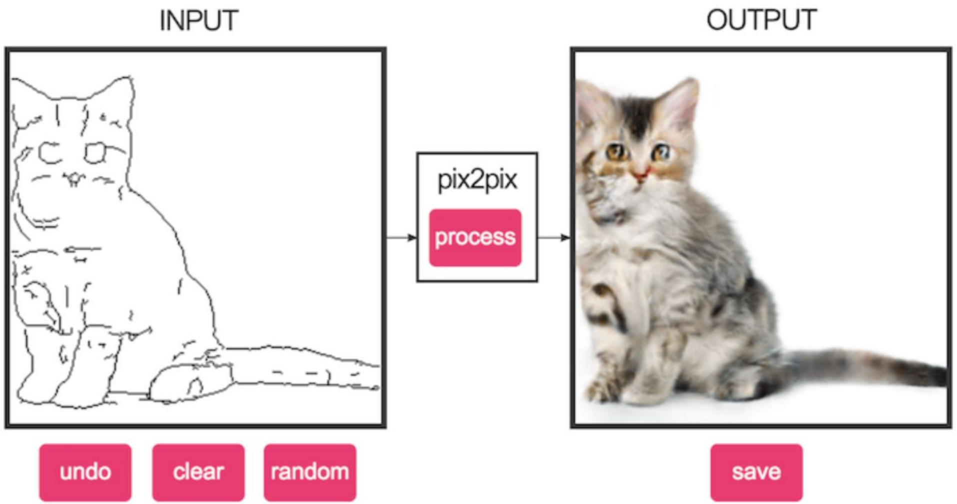
https://missinglink.ai/guides/computer-vision/image-segmentation-deep-learning-methods-applications/

CNN Applications

Self-driving cars

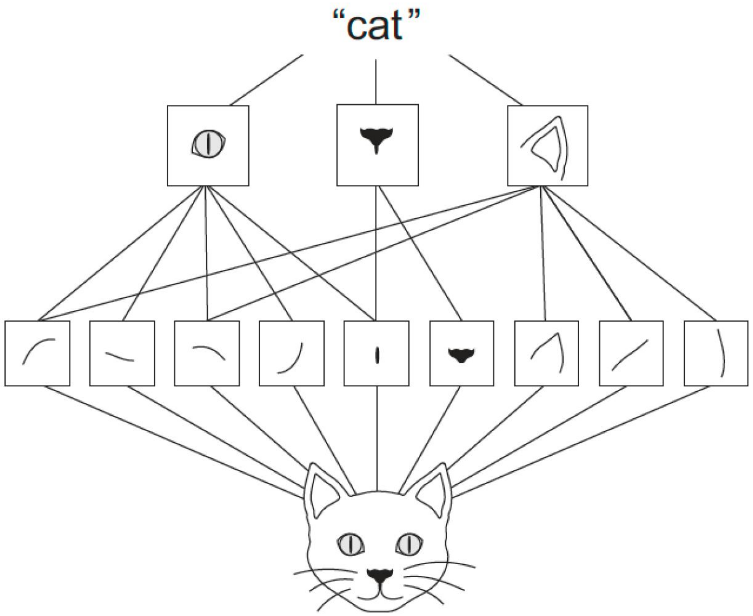
CNN Applications

Generating [realistic pictures from drawn edges]



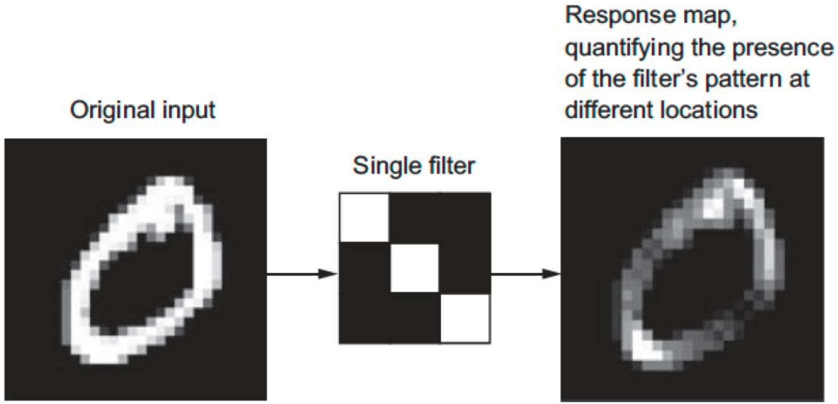
Dense vs Convolutional Layers

- Dense layers learn **global** patterns
- Convolutional layers learn **local** patterns
- CNNs learn **spatial hierarchies of patterns**: one convolutional layer will learn small patterns and the next larger patterns made of the features of the layer before, and so on

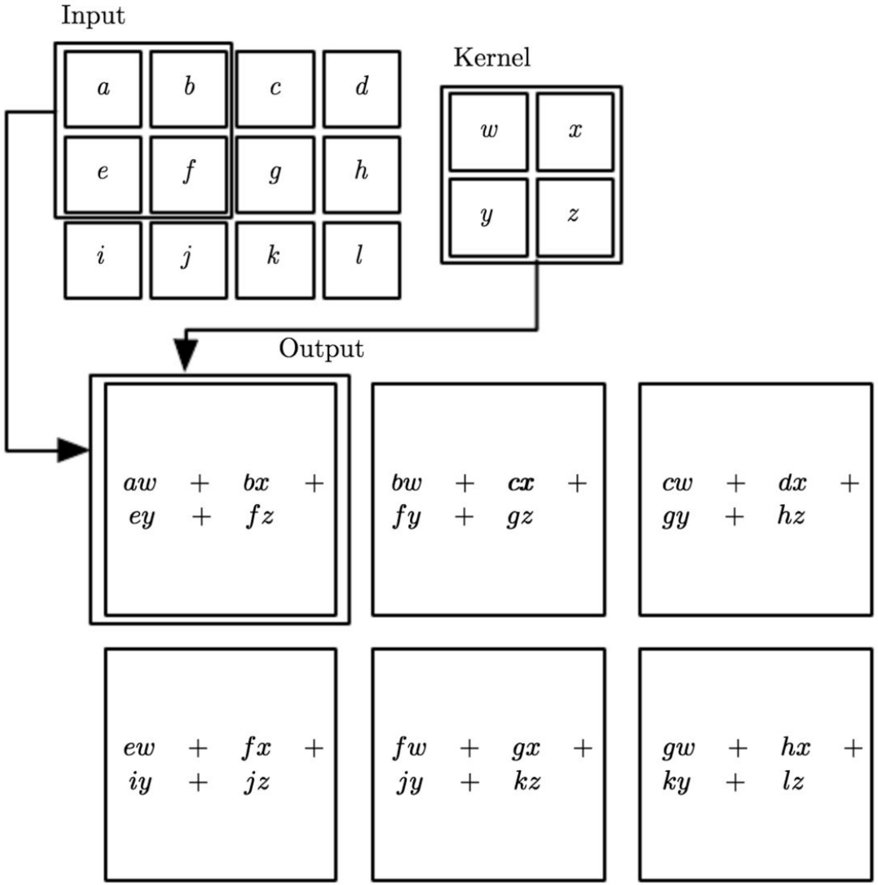


Convolution Layers

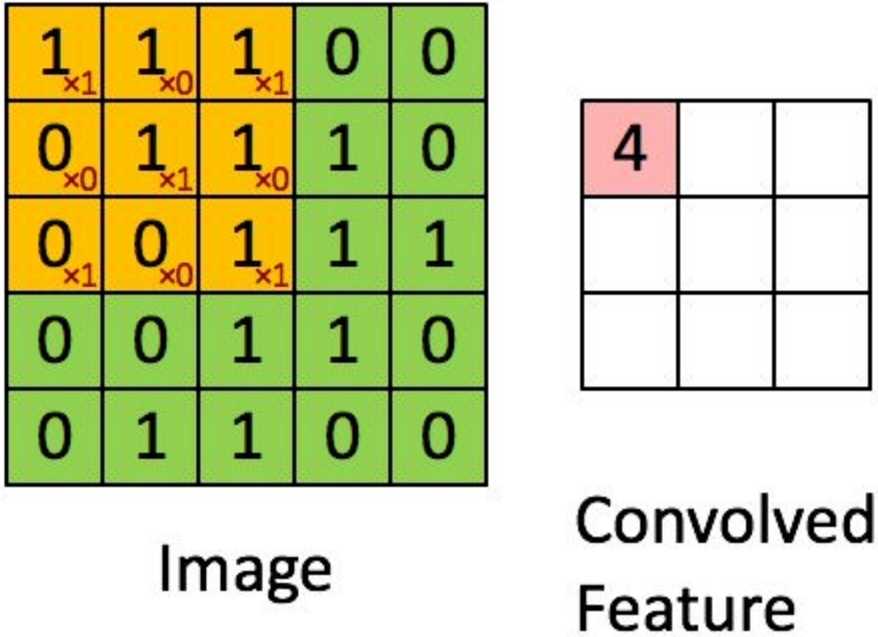
- The primary purpose of the convolution operation is to extract features from the input
- To do this, the input is split into several different areas, and the convolution operation applied to each area
- A summary value is then calculated and kept as part of the output
- We need the input image (sometimes called **feature map**), and a two-dimensional **kernel**, or more commonly, **filter**, which is a weighting function
- The output resulting from this operation is called the **response map**



The Convolution Operation



The Convolution Operation



Filters

- The most common filter sizes are 3 x 3 and 5 x 5
- Sometimes 7 x 7 filters are also used
- A 1 x 1 filter is a special case that we will see later in the course when talking about RNNs
- Odd dimension filters are preferred for computer vision
    - Provide natural padding (we'll see this in the following slides)
    - Ensures a central position or pixel of the filter

Convolution

- Convolution leverages 3 important ideas that can help improve a machine learning system:
  - **Sparse interactions** / sparse connectivity / sparse weights
    - Filters are smaller than the input images and thus fewer parameters (weights) need to be stored
    - This reduces computational expense and improves statistical efficiency
  - **Parameter sharing**
    - The same parameter is used for more than one function in the model
    - In a CNN, each each element of the filter is applied to every position of the input

Convolution

- **Equivariant representations**
  - Parameter sharing causes equivariance to translation: if the input changes, the output changes in the same way
  - A function f(x) is invariant to a function g if f(g(x)) = g(f(x))
  - When processing time-series data, this means that convolution produces a sort of timeline that shows when different features appear in the input
  - Note that convolution is not equivariant to some other transformations, such as changes in the scale or rotation of an image

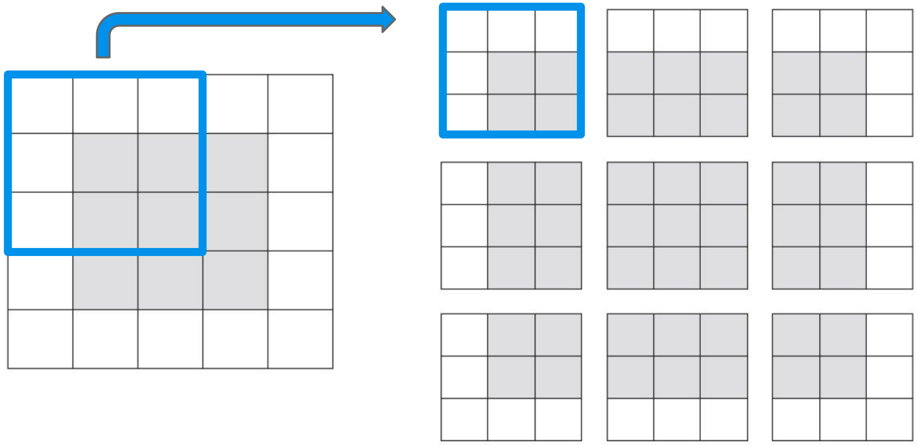
- **Translation invariant**: After learning a certain pattern from one part of an image, it can recognize it anywhere
- Convolution also provides a way to work with inputs of different size

Padding

- Applying a filter to an input image shrinks it - the output dimensions are smaller than the input dimensions
- Additionally, when we perform the convolution operation on an input, the pixels on the edges aren't used as much as the pixels in the middle
- To make the amount of information used more equal across pixels, you can use **padding**
- Padding consists of adding an appropriate number of rows and columns to each side of the image (think a border of pixels)
- **This enables an output with the same dimensions as the input**

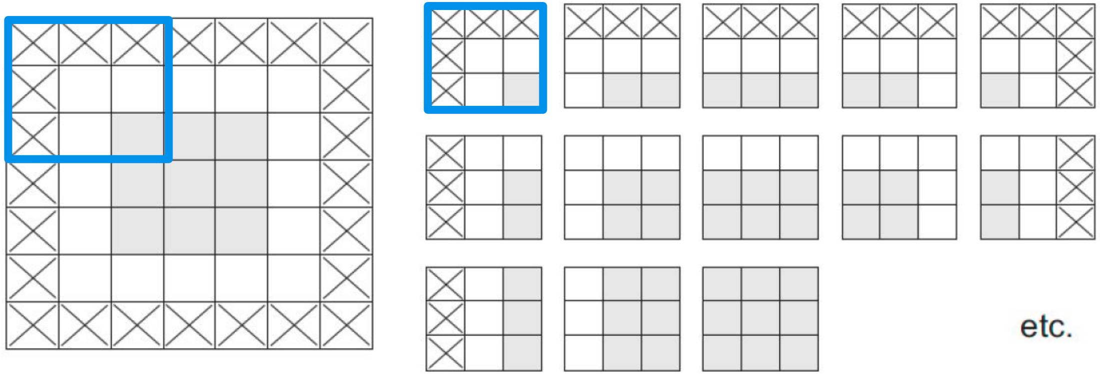
Valid Padding

- In Keras, the default is no padding, or **“valid” padding**
- This means the output will not be the same dimension as the input, and will instead depend on the dimension of the input and the size of the filter
- On the right is a 5 x 5 image
- If we apply a 3 x 3 filter, the output will also be 3 x3



Same Padding

- On the right is the same 5 x 5 image, but with an added border
- If we use a 3 x 3 filter, we need to add $p = 1$ padding - here, p is the number of rows to add to the border of an image
- The output will then be 5 x 5
- Because the input and output have the same dimensions, this is called **“same” padding**



General Padding Formula

- There is a general formula that can help you decide how much padding you need or want
  - Let the input be *n* x *m* and the filter *f* x *f*
  - Without padding, the output would be:

$$(n - f + 1) \times (m - f + 1)$$

- For an output with the same dimension as the input, need *p* such that:

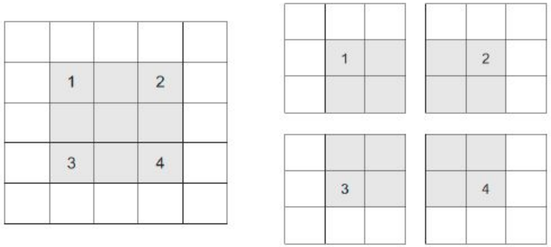
$$n \times m = (n + 2p - f + 1) \times (m + 2p - f + 1)$$

which boils down to $p = \frac{f - 1}{2}$

Convolutional Strides

- Another factor that can influence output size is convolution **strides**
- So far we have assumed that we slide the filter over a single space to extract a new patch - but what if we wanted to slide over 2 spaces, 3 spaces, or more?
- Convolutional strides are convolutions with a stride greater than 1
- If your stride is set equal to 2, you will downsample the width and height of the input image by a factor of 2
- If *s* is the size of the stride, the output will have dimensions

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{m+2p-f}{s} + 1 \right\rfloor$$



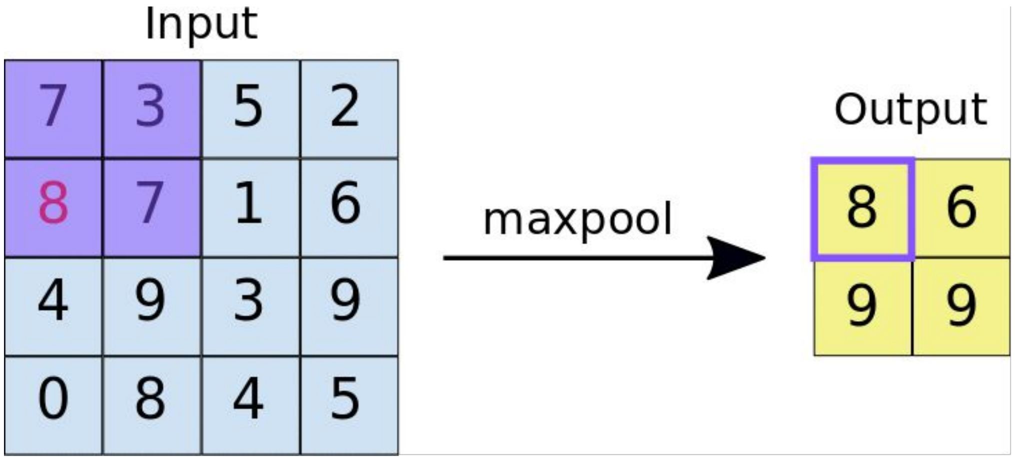
Pooling

- Strided convolutions are rarely used in practice - **max-pooling** is more common
- A typical layer of a CNN consists of 3 stages:
  - **Convolution stage** - linear transformation of the input
  - **Detector stage** - Nonlinear activation (ex: relu activation function is applied)
  - **Pooling stage** - further modification (downsizing) of the output
- A pooling function replaces the output at a certain location with a summary statistic of the nearby outputs
- Pooling greatly reduces the computational expense of the network by decreasing the number of parameters to be learned

Pooling

- There are different types of pooling
  - **Max Pooling**
    - Outputs the maximum value from a patch for each channel
    - Similar to convolution, but instead of transforming patches via a learned linear transformation, they're transformed via a hardcoded max tensor operation
    - Very common
    - Usually done with 2 x 2 windows
  - **Average Pooling**
  - **Weighted Pooling**
    - Based on the distance from the central pixel
- Max pooling tends to work better than other pooling functions and convolutional strides

Max Pooling



Average Pooling

Average Pooling (kernel: (2, 2), stride: (2, 2), padding: (0, 0))

9	6	2	6	8	9
7	9	0	9	9	2
5	5	5	8	5	9
9	4	9	7	2	1
6	6	9	3	4	6
9	3	4	1	7	2

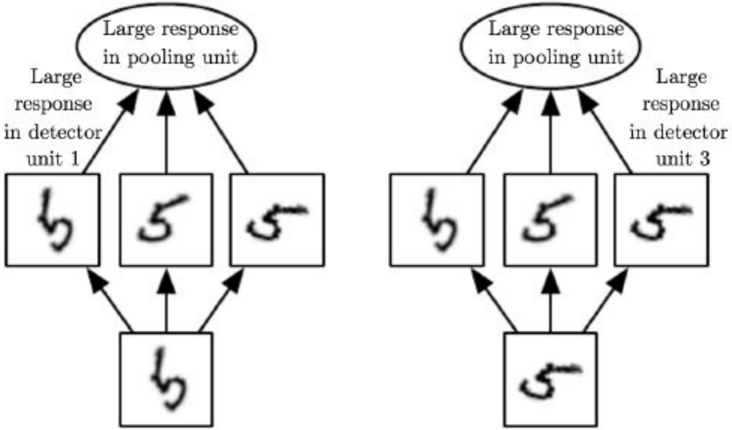
Input

7.75		

Output

Pooling and Invariance

- A pooling unit that pools over multiple features that are learned with separate parameters can learn to be invariant to transformations of the input
- Here, a set of 3 learned filters and a max pooling unit can learn to become invariant to rotation



Terminology

- Convolutions operate over 3D tensors with two spatial axes, **height** and **width**, as well as a **depth** axis (or **channels** axis)
- For a color (RGB) image, the depth is equal to 3
- For black and white (grayscale) images, the depth is equal to 1
- The convolution operation extracts different **patches** from the input image and applies the same transformation to to each of them, resulting in a response map that is also a 3D tensor

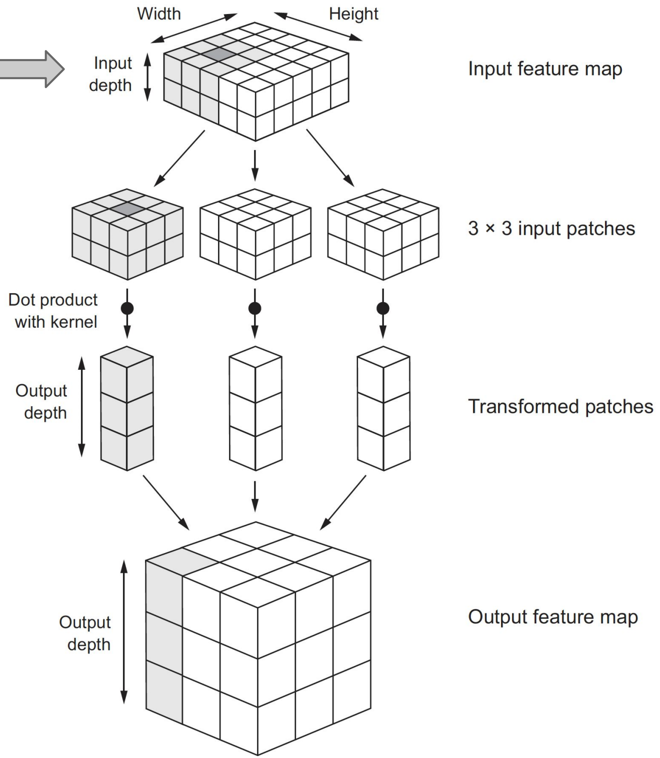
Terminology

- The response map has a width, height, and depth, all of which depend on the input image, filter, padding and stride
- Convolutions are defined by 2 key parameters:
  - Size of the filter
  - Depth of the output response map, i.e., how many filters are applied to the input
- Convolution works by sliding the filter over the 3D input image, stopping at every possible location, and extracting the 3D patch at each location
  - Each 3D patch is transformed into a 1D vector
  - All 1D vectors are then reassembled into a 3D output map

Convolution Schematic

Input image has shape 5x5x2

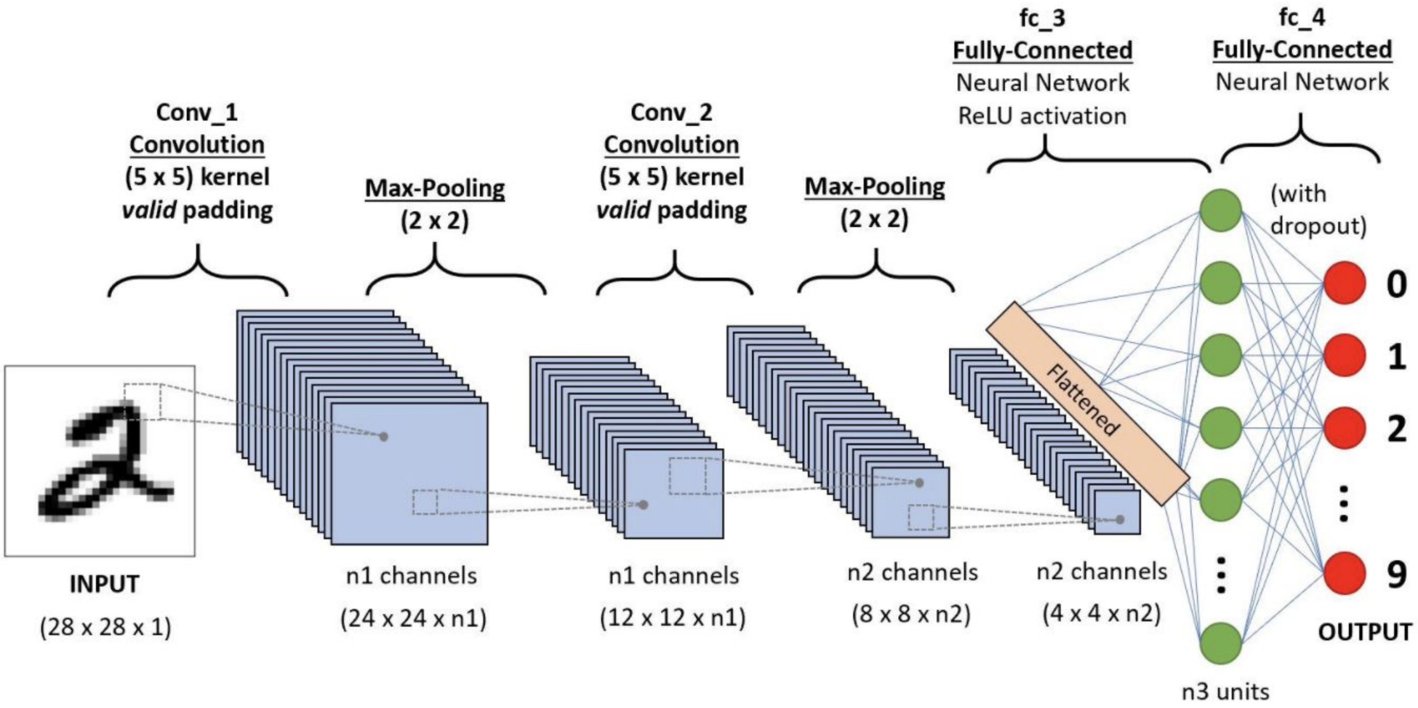
There are 3 filters (not shown) which results in a depth of 3 for each patch.



Here, each filter has a shape of 3x3x2, which results in a total of 9 patches from the original input. Only 3 of those patches are shown here.

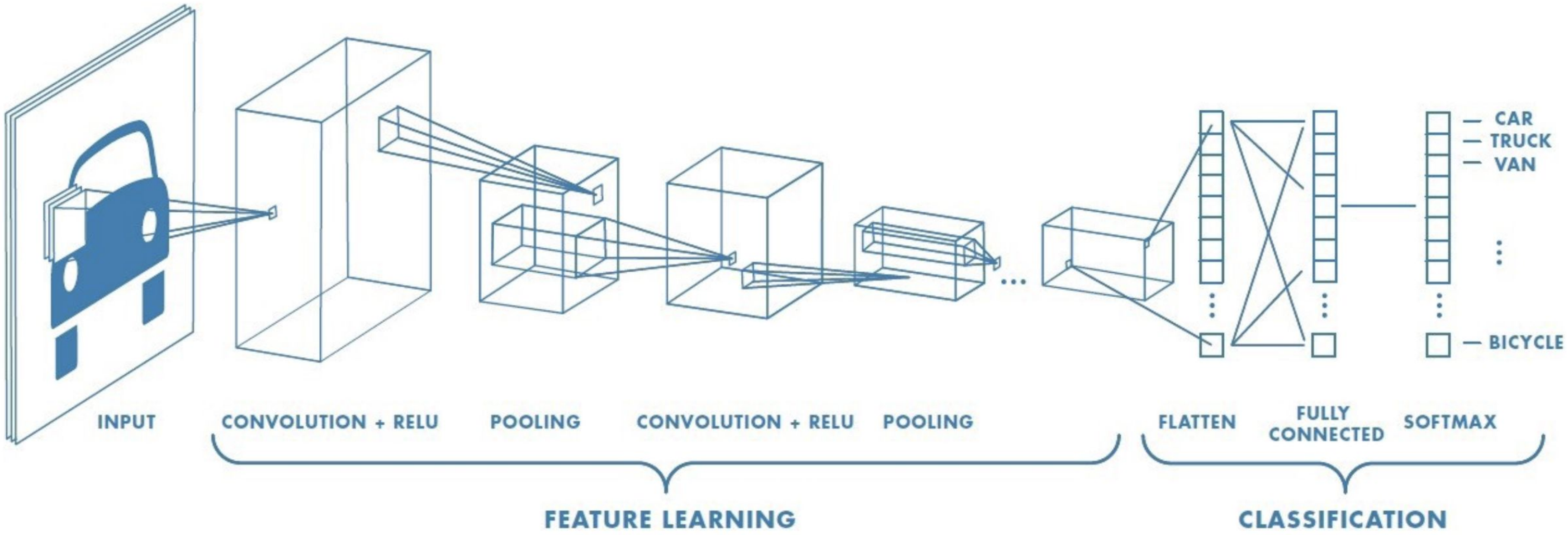
9 total patches, each with a depth of 3 (one number for each filter), resulting in an output shape of 3x3x3.

CNN Schematic



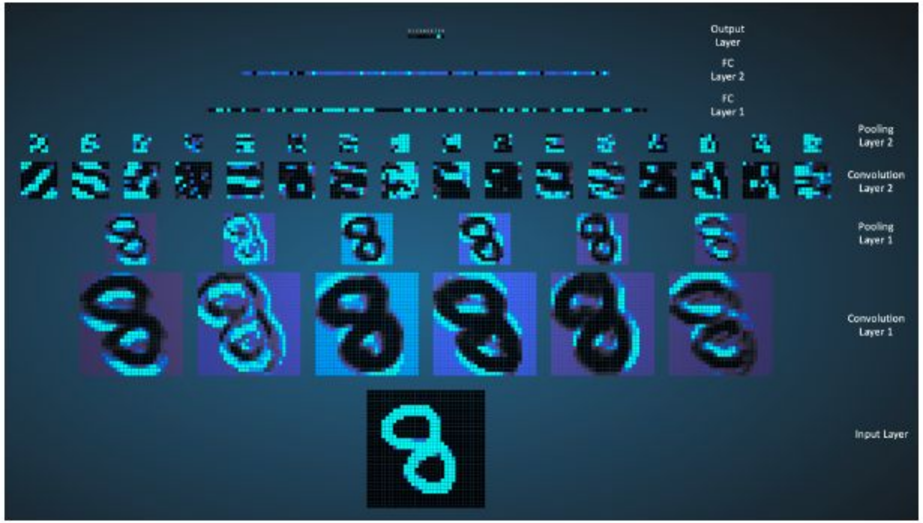
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN Schematic



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

MNIST Example



MNIST Example

Colab notebook

MNIST Example

```
# Define model
model = tf.keras.models.Sequential([
  # Convolutional layer with 32 filters that are 3x3, relu activation function
  tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
  # Pooling layer with 2x2 windows
  tf.keras.layers.MaxPooling2D((2, 2)),

  # Convolutional layer with 64 filters that are 3x3, relu activation function
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
  # Pooling layer with 2x2 windows
  tf.keras.layers.MaxPooling2D((2, 2)),

  # Convolutional layer with 64 filters that are 3x3, relu activation function
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

  # Collapse the 3D tensor
  tf.keras.layers.Flatten(),

  # Fully connected layer with 64 hidden units, relu activation function
  tf.keras.layers.Dense(64, activation='relu'),

  # Softmax output function with 10 classes
  tf.keras.layers.Dense(10, activation='softmax')
])
```