EECE-6822 Machine Learning

CNNs

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

PART 1
Convolutional Neural Networks (CNNs)

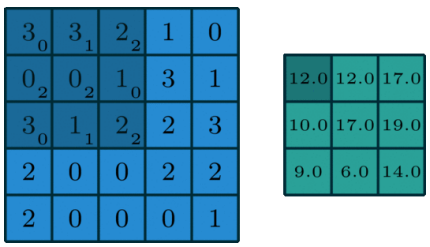
2D Convolutions

- 2D Convolution operation:
  - Start with a **kernel**, which is simply a small matrix of weights
  - Kernel "slides" over the 2D input data
  - Perform an elementwise multiplication with the part of the input it is currently on
  - Sum up the results into a single output pixel
- Kernel repeats this process for every location it slides over
  - Converts 2D matrix of features into another 2D matrix of features
  - Output features are essentially, **weighted sums** (with weights being the values of the kernel itself) of input features located **roughly** in the same location of the output pixel on the input layer

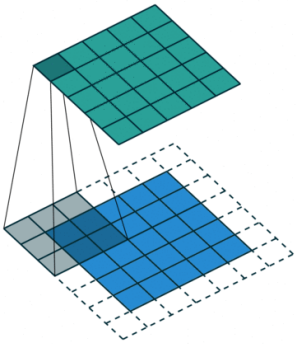
3

3

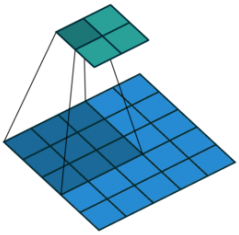
Standard Convolution (1 Channel)



Standard Convolution



With **Padding**



With **Stride 2**

In the 1 channel case: terms filter and kernel are interchangeable

4

4

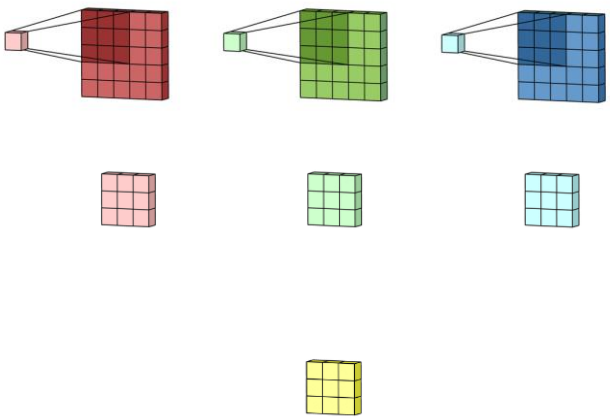
Standard Convolution (3 Channel, RGB)

- In the general case, terms filter and kernel are different.
- Each filter is a collection of kernels
  - One kernel for every single input channel and each kernel being unique.
  - Each of the kernels of the filter “slides” over their respective input channels, producing a processed version of each.
  - Some kernels may have stronger weights than others, to give more emphasis to certain input channels.
  - Each of the per-channel processed versions are then summed together to form one channel.
  - Finally, the bias term - each output filter has one bias term that gets added to the **output channel** to produce the final output channel.

5

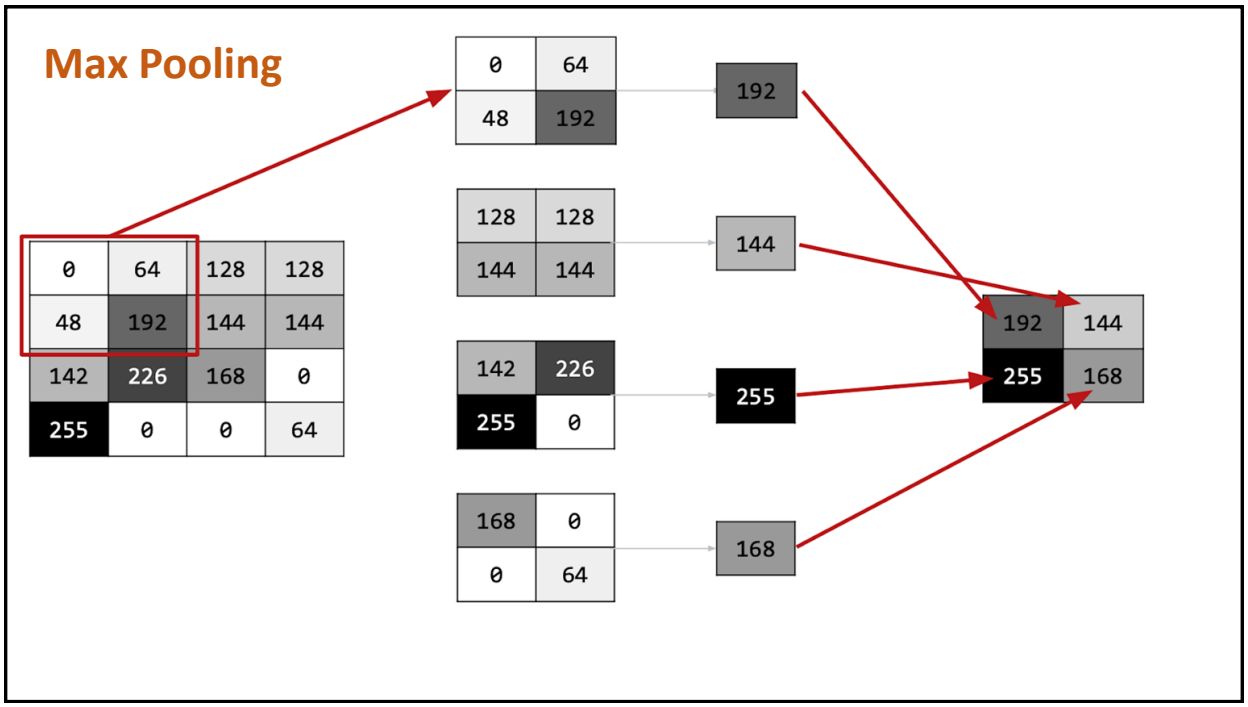
5

Standard Convolution (3 Channel, RGB)



https://medium.com/data-science/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

6



7

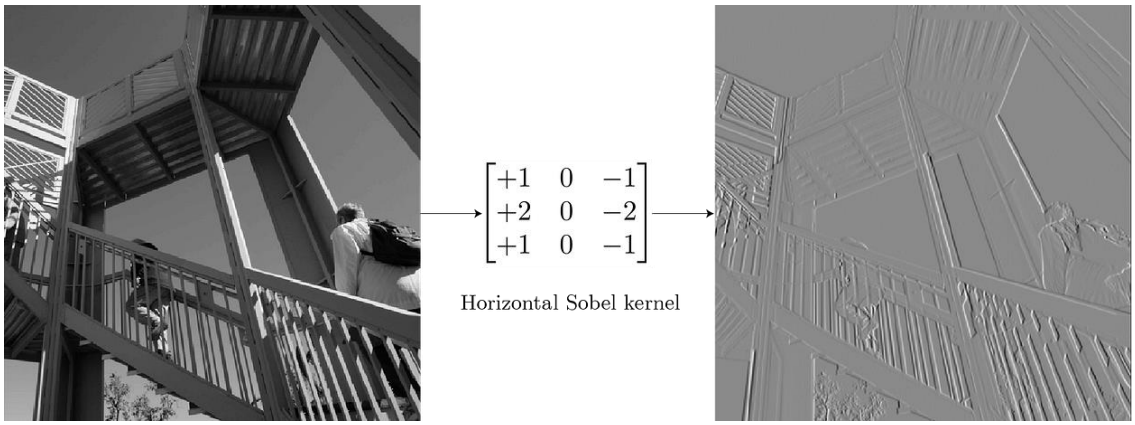
Locality

- Kernel is applied globally across the whole image to produce a matrix of outputs.
- Kernels combine pixels only from a small, local area to form an output.
- Output feature only "sees" input features from a small local area.
- Pixels appear in a consistent order → nearby pixels influence a pixel (e.g., if all nearby pixels are red, it is very likely the pixel is also red)
- If there are deviations: **anomaly that could be converted into a feature** - can be detected from comparing a pixel with its neighbors (other pixels in its locality/vicinity).
- That was the idea of earlier computer vision feature extraction methods. For example, edge detection, one can use a Sobel edge detection filter, a kernel with fixed parameters, operating just like the standard one-channel convolution – **see next slide**.

8

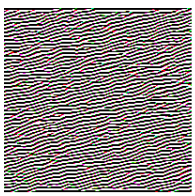
8

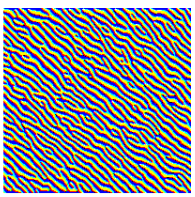
Applying Vertical Edge Detector Kernel



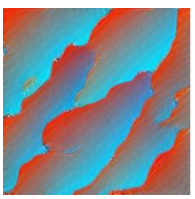
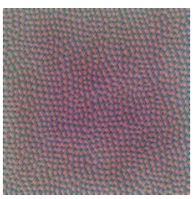
Deep Learning

- **Can useful kernels be learnt?**
- For early layers operating on raw pixels, we expect feature detectors of fairly low-level features, like edges, lines, etc.
- Convolved images are still images - output of a small grid of pixels from the top left of an image will still be on the top left.
- We can run another convolution layer on top of another to extract deeper features - which we can visualize:





Feature visualization for 3 different channels from 1st convolution layer of GoogLeNet

Feature Visualization of channel 12 from 2nd and 3rd convolutions

Receptive Field

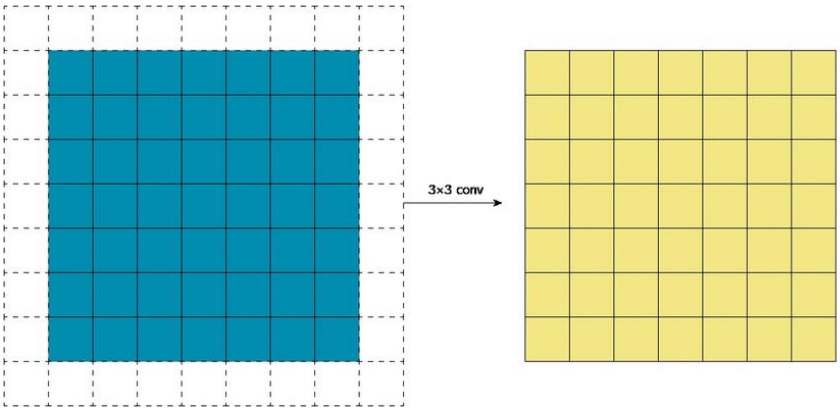
- CNN architecture:
  - Input sizes grow smaller and smaller from start to end of the network
  - Number of channels grow deeper.
  - Achieved through strides and/or pooling layers.
- **Locality** determines what inputs from the previous layer the outputs get to see.
- **Receptive field** determines what area of the original input to the entire network the output gets to see.

11

11

Strided Convolution

We only process windows a fixed distance apart, while skipping the ones in the middle; i.e., we only keep outputs a fixed distance apart while removing the rest.

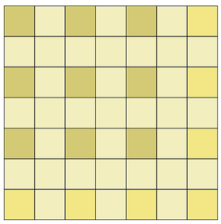


12

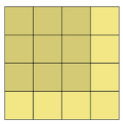
12

Convolution on Top of Strided Convolution
Where Things Get Interesting

- We apply a nonlinearity to the output of a strided convolution
- Then, stack another new convolution layer on top – **things get interesting!**
- Even if we apply a kernel of the same size (3×3) to the output of the strided convolution, the kernel would have a **larger effective receptive field:**



Operation on the output with the removed pixels

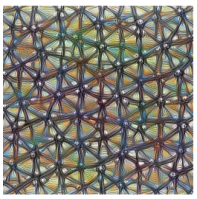


Operation on the true strided output

- That is because output of strided layer still does represent the same image
- Each single pixel in the output is a "representative" of a larger from the same rough location from the original input
- When next layer's kernel operates on the output, it is operating on pixels collected from a **larger area**

13

mixed3a, channel 31



mixed4a, channel 11

mixed5a, channel 14

- Expansion of receptive field allows convolution layers to combine the low-level features (lines, edges) into higher-level features (curves, textures) – see **mixed3a** layer
- Followed by a pooling/strided layer, the network continues to create detectors for even higher-level features (parts, patterns) – see **mixed4a** layer.
- Repeated reduction in image size across the network results in input sizes of just 7×7 (by 5th block/layer of convolutions) compared to inputs of 224×224. Here, each single pixel represents a grid of 32×32 pixels - which is huge.
- Compared to earlier layers, where an activation meant detecting an edge, here, an activation on the tiny 7×7 grid is one for a very high-level feature, such as for birds.

14

CNN

- Starts by learning very low-level feature detectors
- As its receptive field is expanded across layers, it learns to combine those low-level features into progressively higher-level features: not an abstract combination of every single pixel, but rather, a strong visual hierarchy of concepts.
- It eventually detects entire visual concepts such as faces, birds, trees, etc., and that is what makes CNNs such powerful, yet efficient with image data.

15

15

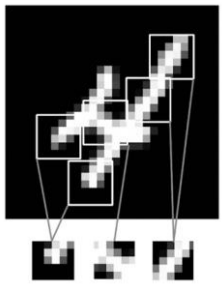
Image Classification With Convolutions

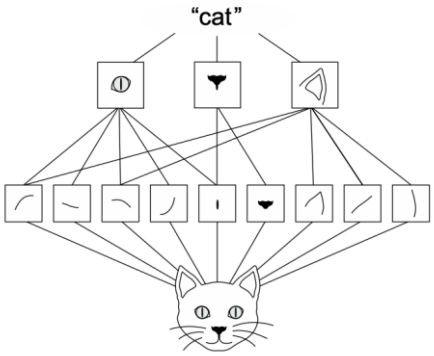
16

16

The Convolution Operation

*"The fundamental difference between a densely connected layer and a convolution layer is this: **Dense layers learn global patterns** in their input feature space (for example, for an MNIST digit, patterns involving all pixels), whereas **Convolution layers learn local patterns**—in the case of images, patterns found in small 2D windows of the inputs In the previous example, these windows were all 3 × 3."*





"They can learn spatial hierarchies of patterns. A first convolution layer will learn small local patterns such as edges, a second convolution layer will learn larger patterns made of the features of the first layers, and so on."

"Deep Learning with Python" by François Chollet

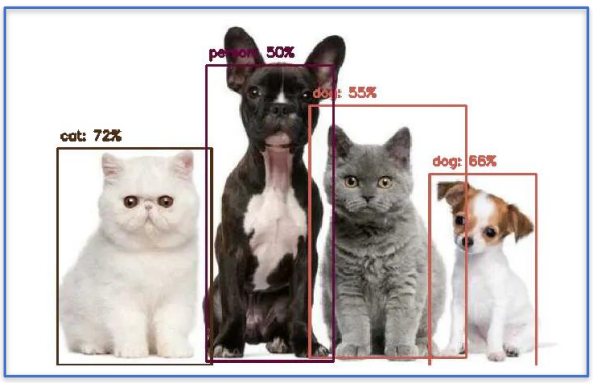
17

Computer Vision Main Problem Types

Image Classification
(**Multi-Class** Classification)

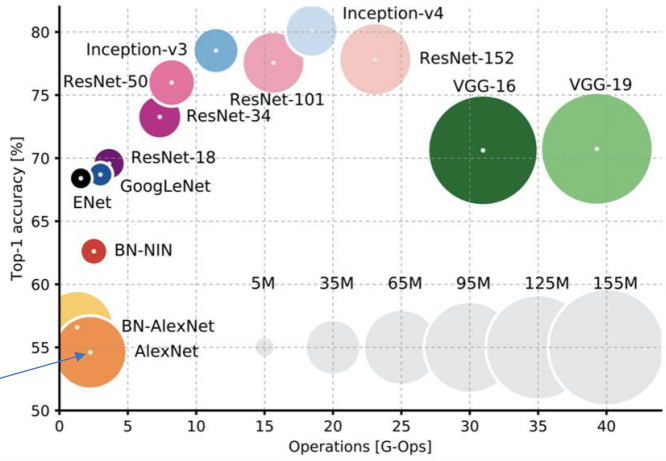
Cat: 70% Dog: 80%

Object Detection
Multi-Label Classification + Object **Localization**



18

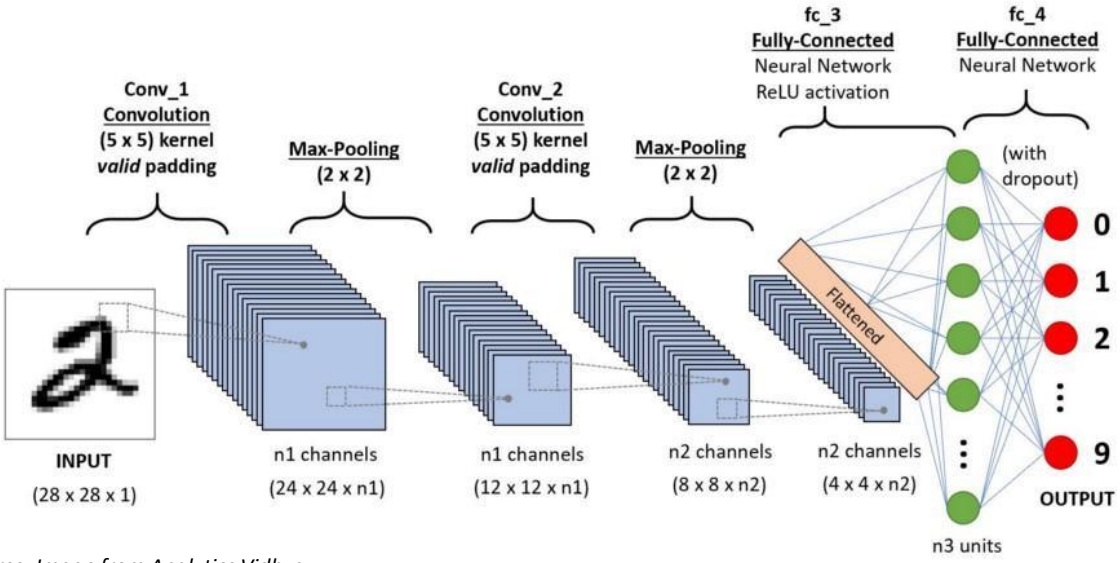
Model **Evolution**



(2012)

19

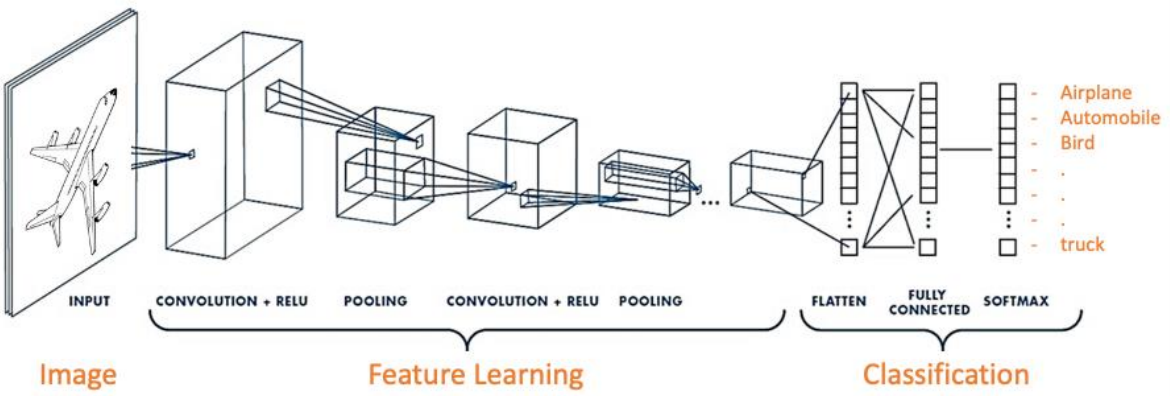
Image Classification with CNNs (Dataset: MNIST, Digits 0-9)



Source: Image from Analytics Vidhya

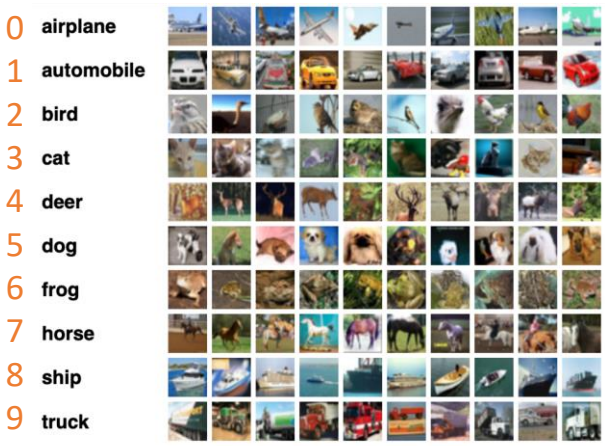
20

Image Classification with CNNs (Dataset: CIFAR-10, 10 Objects)



21

CIFAR-10 Dataset



https://www.tensorflow.org/datasets/catalog/cifar10

22

Why use a CNN instead of a DNN

- **DNN**
  - We saw how to build Neural Networks that classify images of digits (MNIST).
  - MNIST images are simple (28x28 pixels monochrome images), containing just the object centered in the image, with no background
- **CNN**
  - CIFAR dataset contains 32x32 color images (3 color channels)
  - Images are not centered and can have the object with a background (such as airplanes that might have a cloudy sky behind)
  - We will use CNNs to recognize the 10 classes of CIFAR ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship' and 'truck').
- For tasks like image recognition, CNNs excel at finding spatial patterns through local connectivity and weight sharing
  - Leads to better generalization and reduced overfitting
- CNNs are more efficient for high-dimensional data like images
  - Automate feature extraction
  - Reduce number of trainable parameters compared to fully connected DNNs

23

Image Classification
Training a Model - Transfer Learning

24

24

Training Pipeline: **Need Lots of Data**

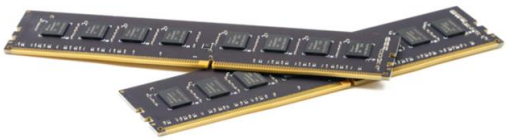
1000 Classes

1000 Images / Class

25

Training Pipeline: **Need Compute Resources**

Computationally Intensive
Repeated Many Times (Epochs)



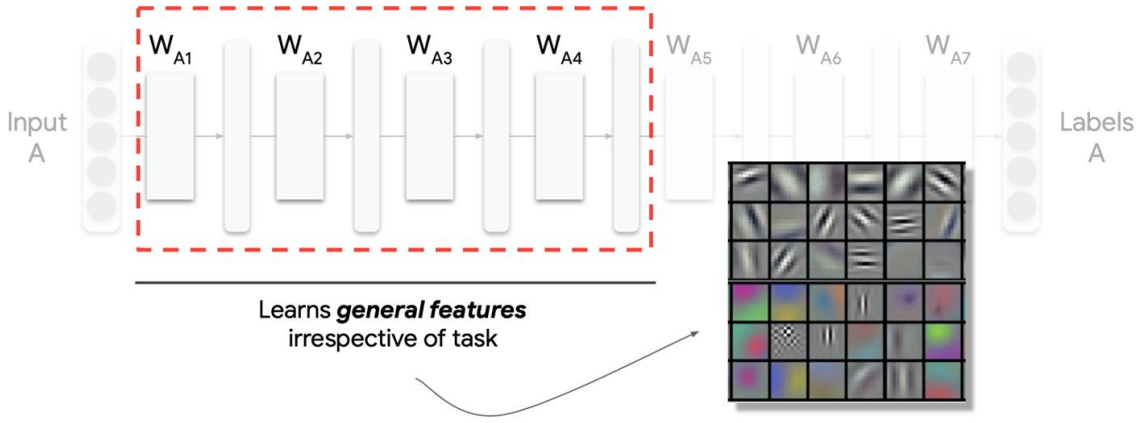
Memory

Compute

GPU and
Accelerators

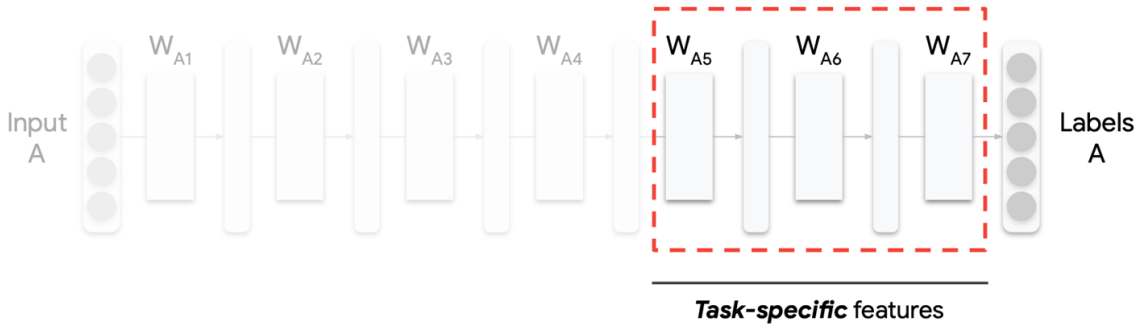
26

End Result of Training



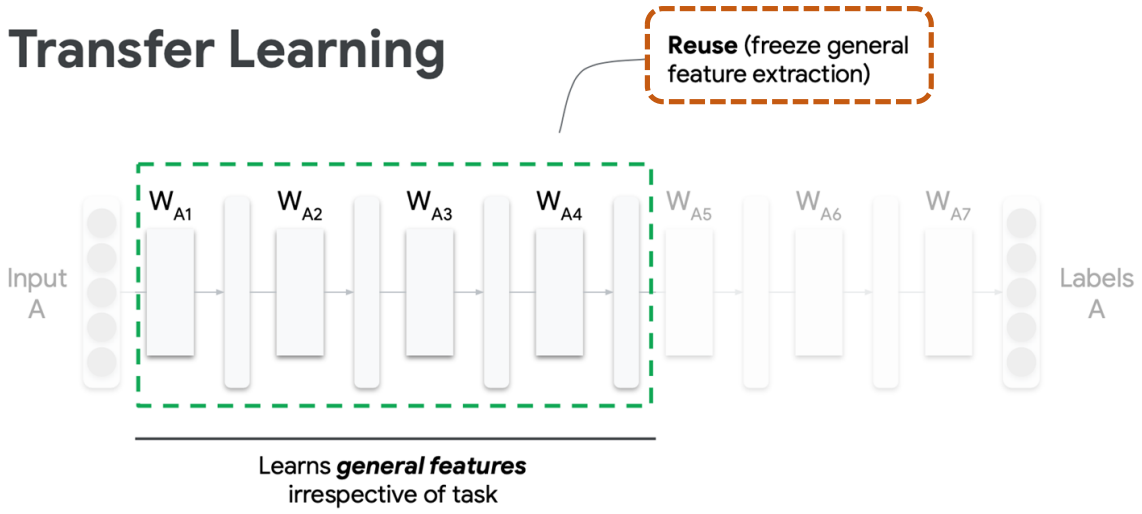
27

End Result of Training



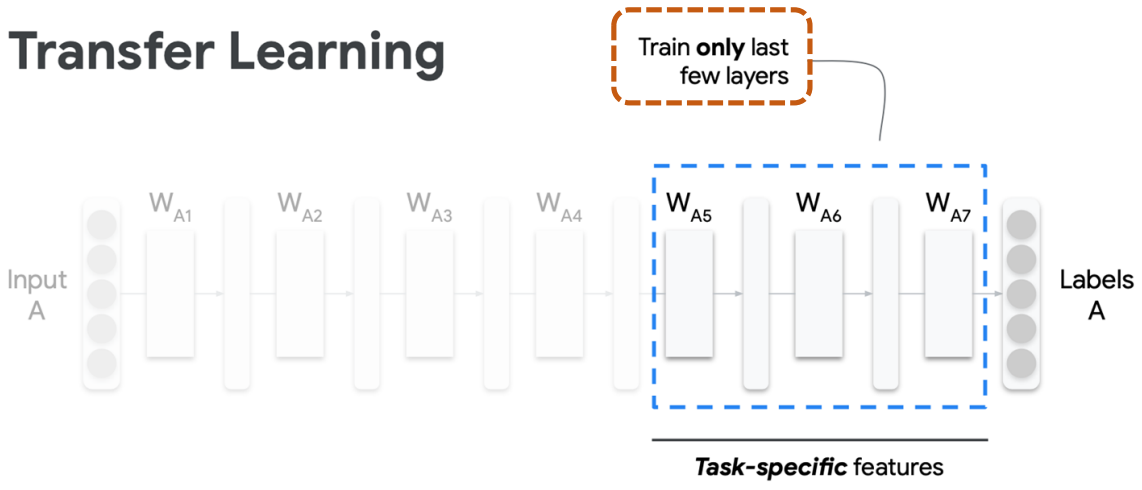
28

Transfer Learning



29

Transfer Learning



30

PART 2

Code Time!

31

31

Code Time

- See demonstration and discussion in class.
- See also links in the “Code Examples” for this Lecture.
- CNN_Cifar_10.ipynb
- Cat_Dog_Detection_using_Transfer_Learning.ipynb

32

32

Conclusion

Takeaways

Why Use CNNs

- For tasks like image recognition, CNNs excel at finding spatial patterns through local connectivity and weight sharing
  - Leads to better generalization and reduced overfitting
- CNNs are more efficient for high-dimensional data like images
  - Automate feature extraction
  - Reduce number of trainable parameters compared to fully connected DNNs

References and Credits

Many of the teaching materials for this course have been adapted from various sources. We are very grateful and thank the following professors, researchers, and practitioners for sharing their teaching materials (in no particular order):

- Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. https://amlbook.com/slides.html
- Ethem Alpaydin. https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/
- Natasha Jaques. https://courses.cs.washington.edu/courses/cse446/25sp/
- Lyle Ungar. https://alliance.seas.upenn.edu/~cis520/dynamic/2022/wiki/index.php?n=Lectures.Lectures
- Aurelien Geron. https://github.com/ageron/handson-ml3
- Sebastian Raschka. https://github.com/rasbt/machine-learning-book
- Trevor Hastie. https://www.statlearning.com/resources-python
- Andrew Ng. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Richard Povineli. https://www.richard.povinelli.org/teaching
- … and many others.

35