Aerial Imagery Classification

OpenAI Carribean Challenge – Driven Data
South America (Colombia)
Cynthia Habonimana

LAST UPDATE | AUGUST 2020

About

CYNTHIA HABONIMANA

Computer Science Graduate

California State University, Long Beach

@buildwithcycy

Click Github icon for Link to Project
Click Twitter icon for Link to my Twitter Profile

LAST UPDATED AUGUST 2020

Presentation Highlights

1. Project Motivation
2. Problem description
3. Proposed Solution
4. Tools needed
5. Next steps

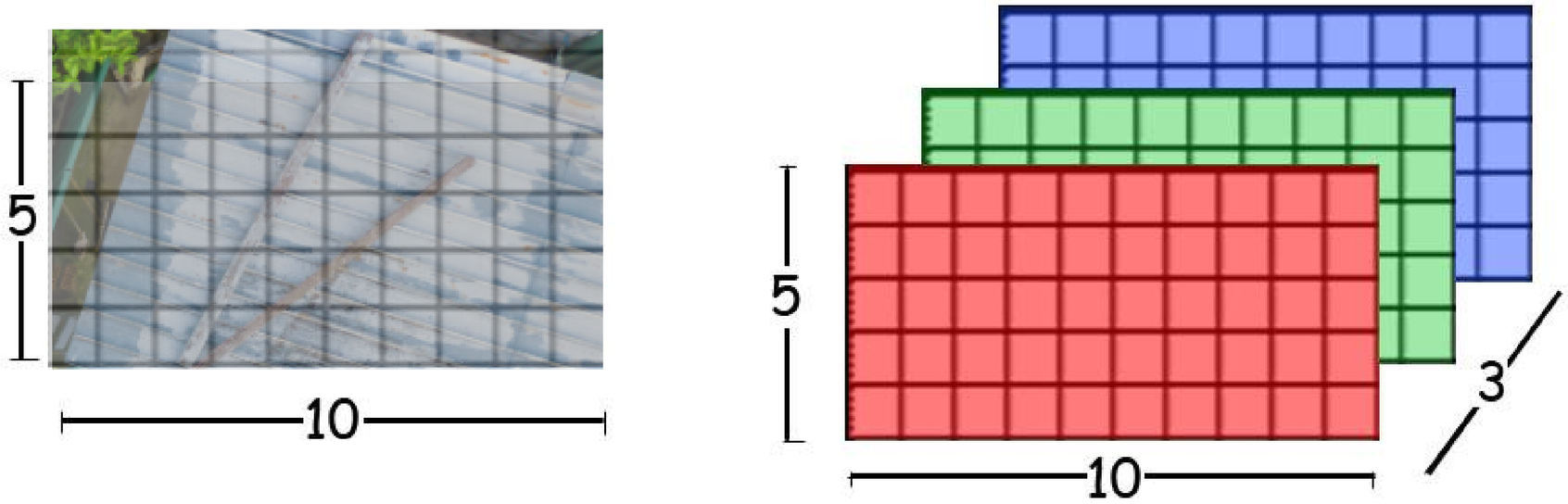
Acknowledgements

Special thanks to Hugo Larochelle for his mentorship and continuous guidance on this project.
The figures in this presentation are from the Deep Learning open source repository by Andrew Glassner.

Project Motivation

- Natural catastrophes result in significant losses in lives and infrastructure. (E.g. $44 billion for recovery were allocated from Hurricane Maria between the U.S. Federal Emergency Management Agency and the U.S. Department of Housing and Urban Development*).
- In emergency situations, it is critical to rapidly identify which houses have a high likelihood of collapsing during a natural catastrophe. This can take several weeks.
- The material of a rooftop can indicate the likelihood of a house to collapse.
- We can explore how to use machine learning to classify rooftop pictures taken by drones and identify houses which are more likely to collapse.

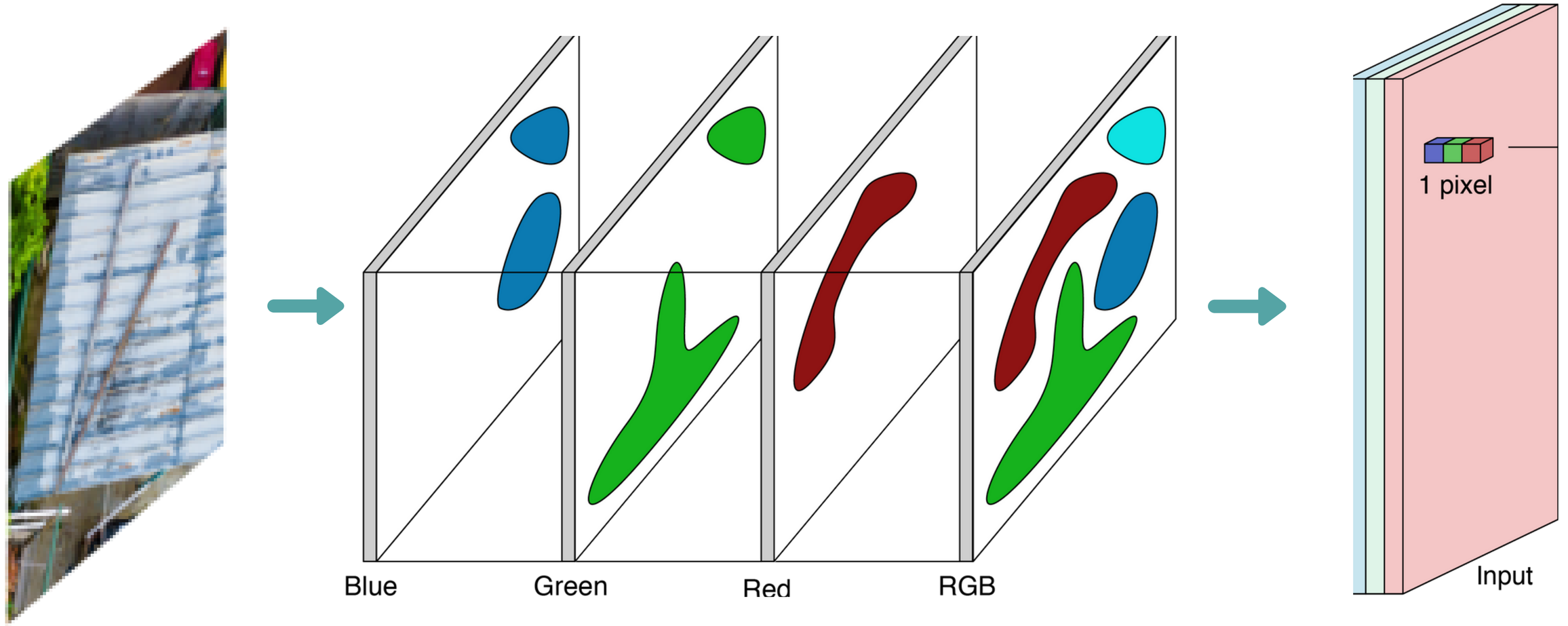
(*) The Washington Post



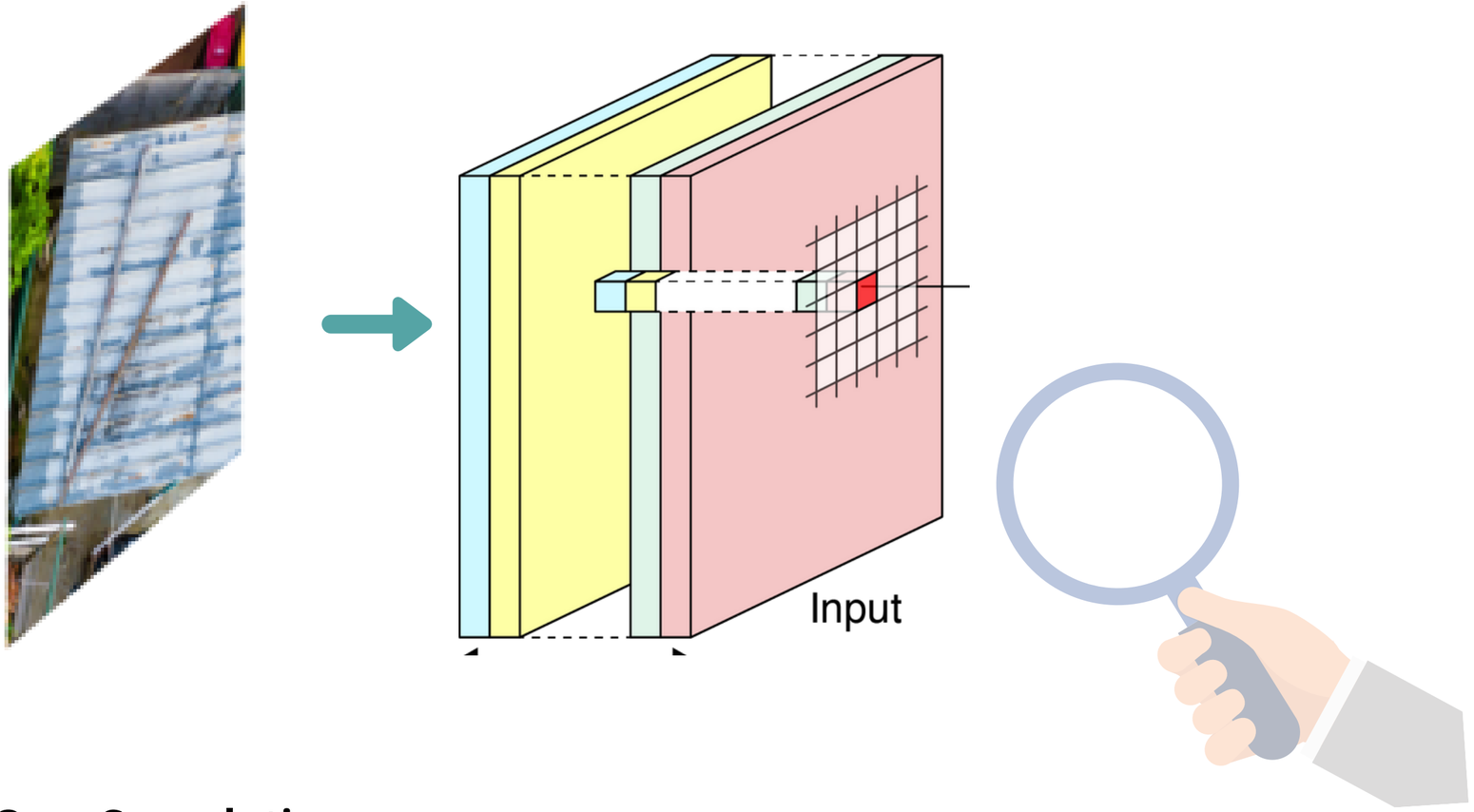
Digital pictures are represented using multiple pixels.

- A pixel is a small square of color
- A pixel represents different intensities of red, green, or blue (RGB).
- Red, Green, and Blue are mixed to generate all the other colors.
- The pixels on a screen are so small and close to each other that the human eye merges them

Topics to Explore: Trichromacy



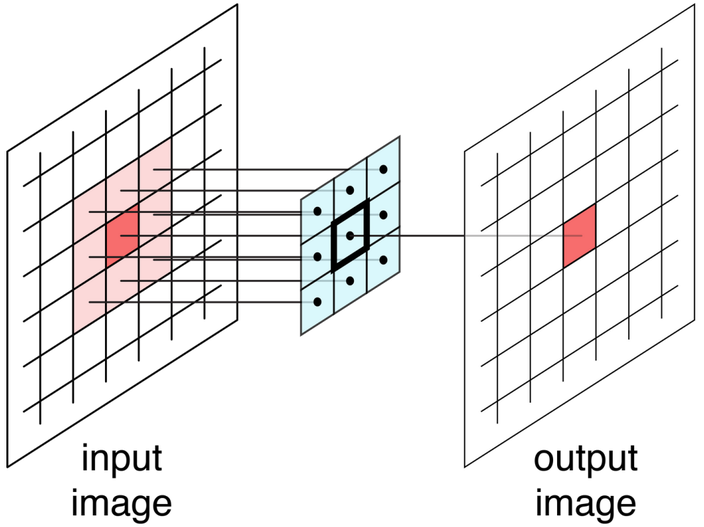
Each input image that is fed to our model is a stack of three layers: red, green, and blue layers.
In our program, the layers are called channels. Each pixel is represented as follows:
Pixel ***x*** **= [Red, Green, Blue] = [channel 0, channel 1, channel 2]**
If the image contains some transparency, we will include a fourth channel *alpha*
Pixel **x = [Red, Green, Blue, Alpha] = [channel 0, channel 1, channel 2, channel 3]**

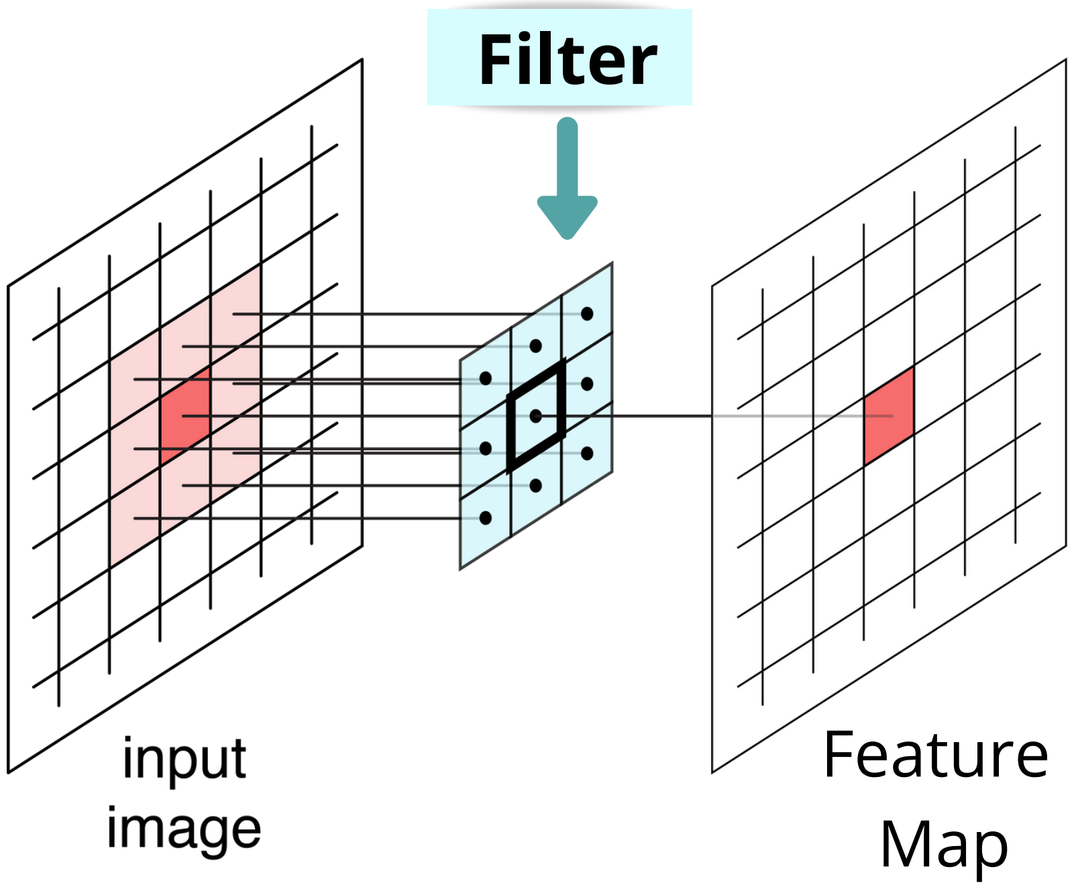


Step One: Convolution

The human brain has distinct regions which can detect edges, shapes, and color to identify what we see. This inspired scientists to build networks which follow a similar concept, convolutional neural networks (CNNs).

In order for our machine learning model to recognize edges, curves, shapes, and special features in a image, we use a set of numbers known as **weights** in a matrix called **filter** (in blue) to ouput a new matrix

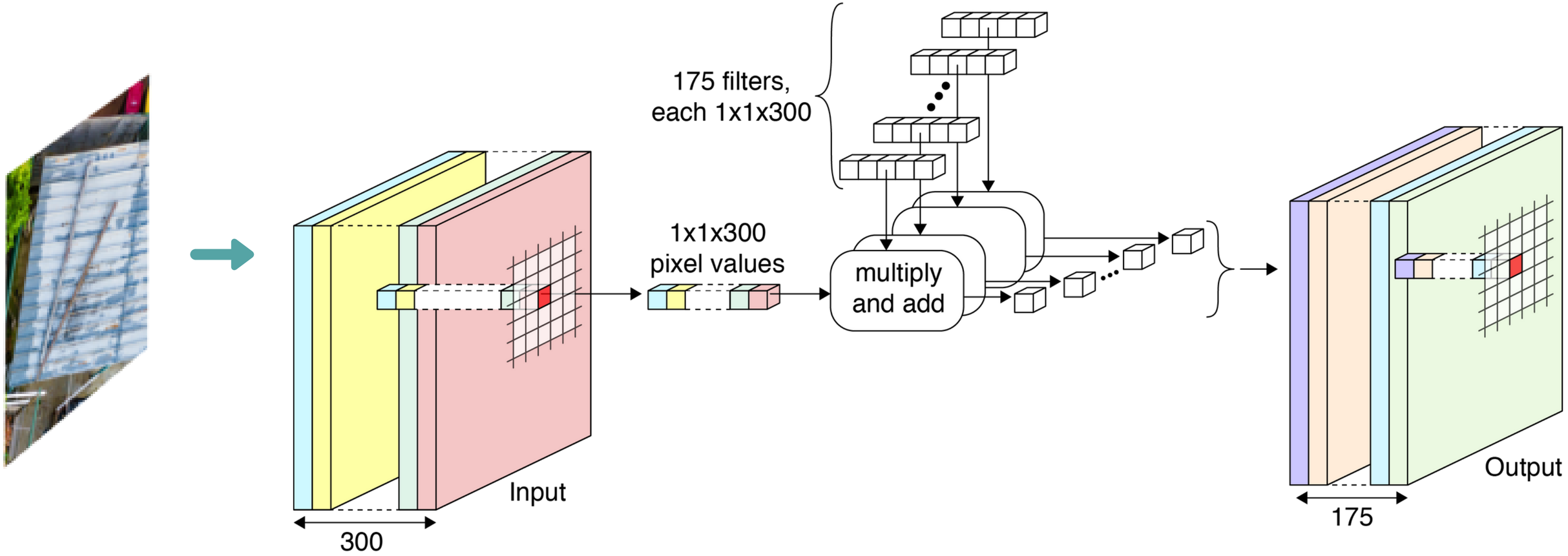




Step One: Convolution

Like a small sliding window, this filter is moved across the input image. At each slide, numbers from the input image and the weights in the filter are multiplied. The products are summed together and the sum is placed in the new matrix. This new matrix contains features that the model judged to be important in learning how to identify the material of a rooftop. To use the proper terminology, we will call the output of one filter applied to the previous layer, a feature map.

This specific process of multiplications and additions to obtain a feature map is called a convolution.
This is repeated until all the numbers from the input image have been multiplied by weights of the filter and added together.



Step One: Convolution

TO-DO

- **Explain shared weights further**
- **Add slides on the other steps (Pooling, Flatening, Full Connecting, Softmax)**
- **Add dimensions for each layer in the graphs below**
- **Add topics to explore**

Regular Steps in Training a CNN

Loading data

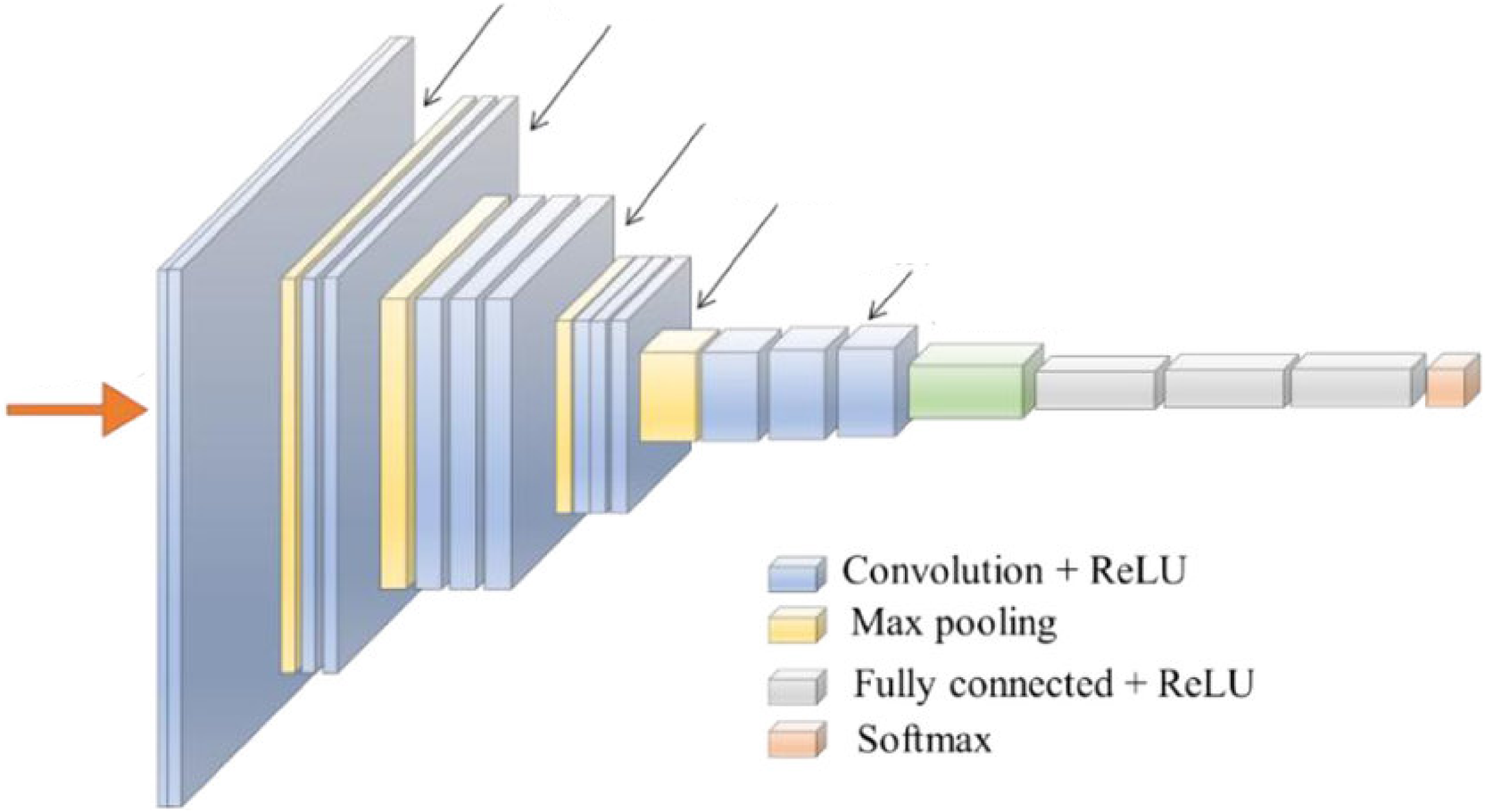
1 Convolution

2 Pooling

3 Flattening

4 Full Connecting

Softmax (Prediction)


Convolution + ReLU
Max pooling
Fully connected + ReLU
Softmax

CONVOLUTIONAL NEURAL NETWORKS

Training a CNN with a twist:

Transfer Learning

Loading data

Base Model: Pre-trained VGG-16

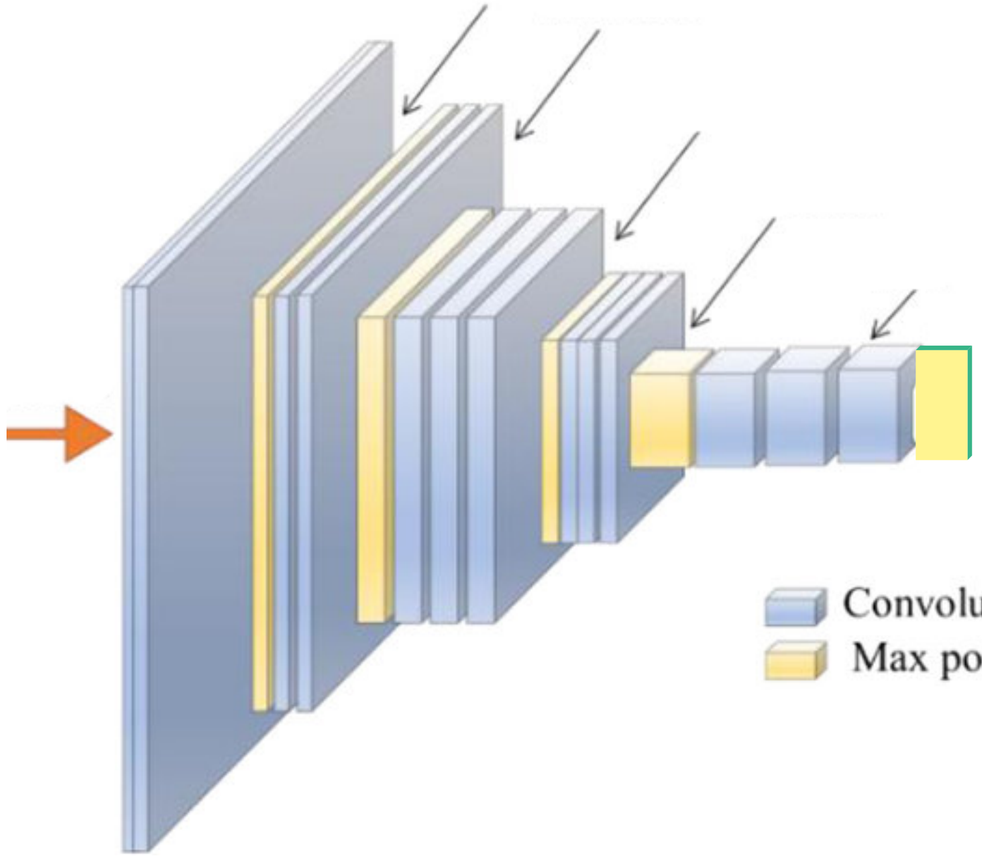
1. Convolution
2. Pooling

Head Model: Custom model

3. Pooling
4. Flattening
5. Full connecting
6. Dropout

} dropout layers added between fully-connected layers

Softmax (Prediction)



Convolution + ReLU
Max pooling

Flattening
Fully Connected + ReLU
Dropout
Output

Base Model
VGG-16

Head Model
Custom Model

Training a CNN with a mega twist:

Using VGG-16 only once

Loading data

Base Model: Pre-trained VGG-16

1. Convolution
2. Pooling

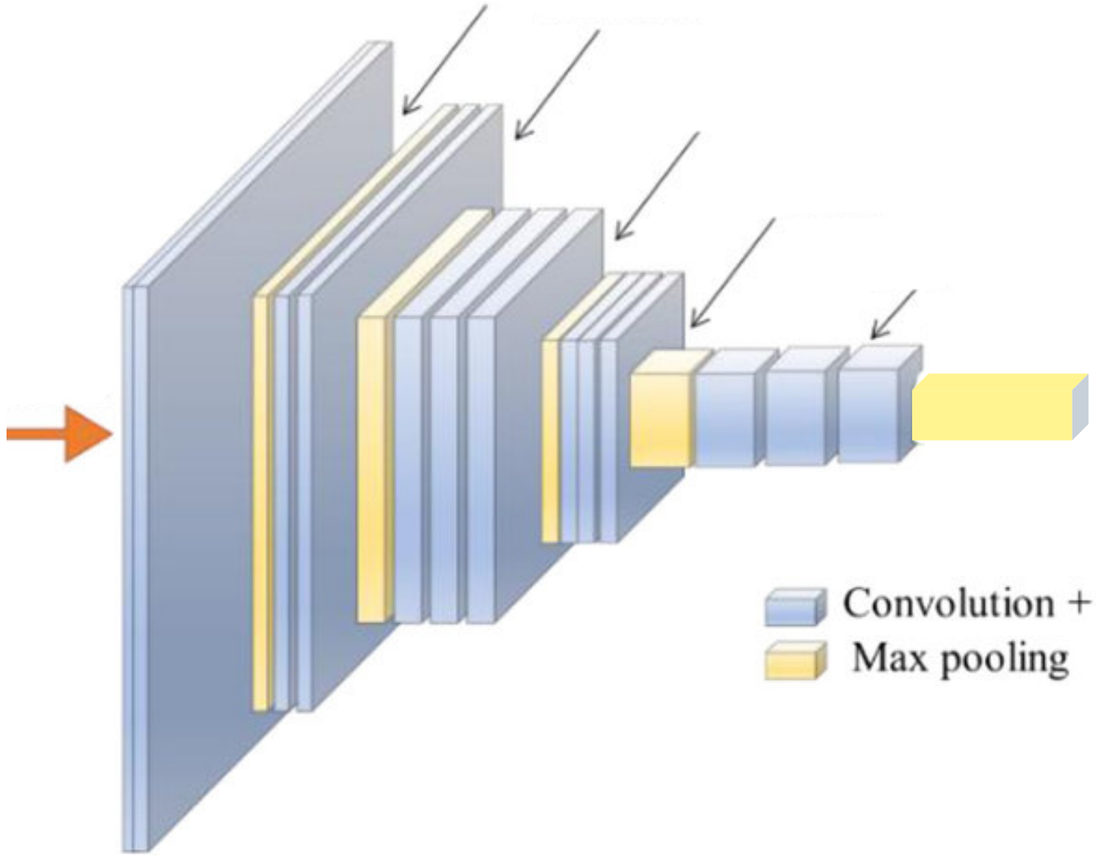
Loading data from the last VGG-16 pooling layer

Custom Model with a new input layer

3. Input layer with Pooling
4. Flattening
5. Full connecting
6. Dropout

dropout layers added between fully-connected layers

Softmax (Prediction)



Input Layer
Flattening
Fully Connected + ReLU
Dropout
Output

Base Model VGG-16

Custom Model with new input Layer