What's happening to the Orangutan?

An image classification study

By Cindy Barrientos

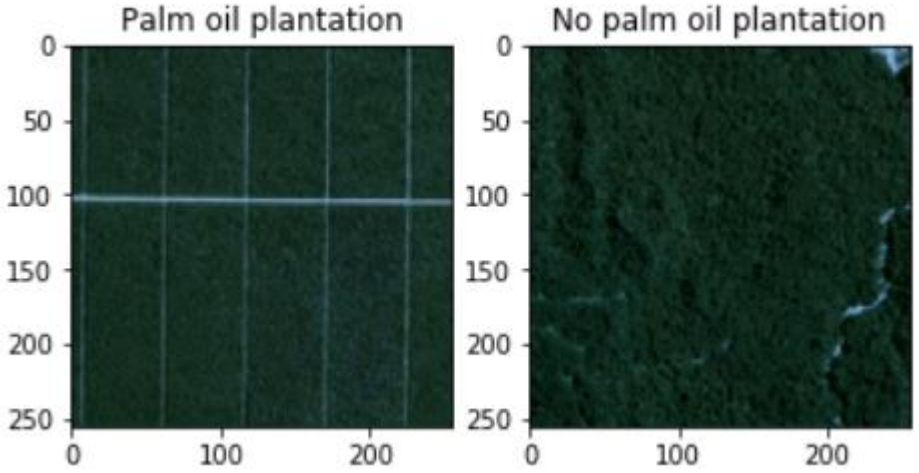
The problem

- Palm oil plantations generate cheap vegetable oils for production
  - Frozen foods, soap, makeup, detergents
- Massive deforestation
  - Orangutans, native to Malaysia and Indonesia, now critically endangered
  - Orangutan population by year
    - 288,500 (1973)
    - 100,000 (2016)
    - 47,000 (2025, estimated)
- Satellite imaging to better document location and distribution of world palm oil plantations

Sources: https://www.rainforest-rescue.org/topics/palm-oil#start
https://www.lifegate.com/people/news/albino-orangutan-rescued-indonesia

Satellite Images

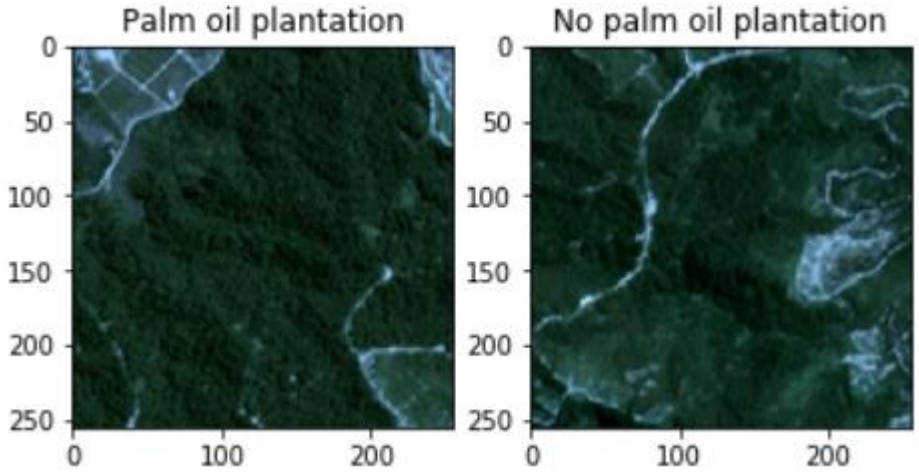
- Satellite imaging to better document location and distribution of world palm oil plantations



Images courtesy of: https://www.planet.com/
and https://www.figure-eight.com/

Satellite Images

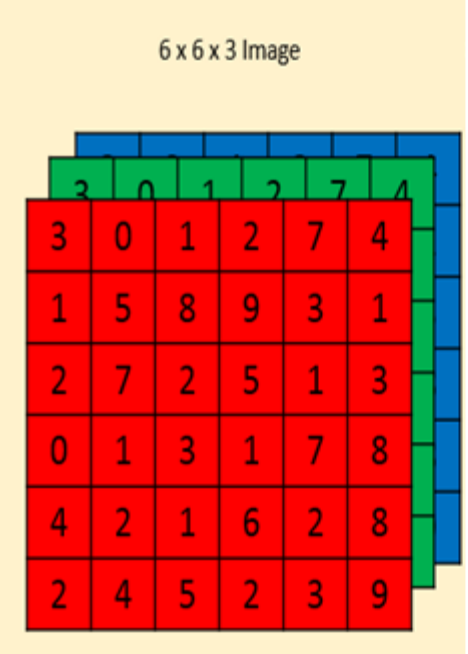
- Satellite imaging to better document location and distribution of world palm oil plantations



Images courtesy of: https://www.planet.com/
and https://www.figure-eight.com/

Convolutional neural networks

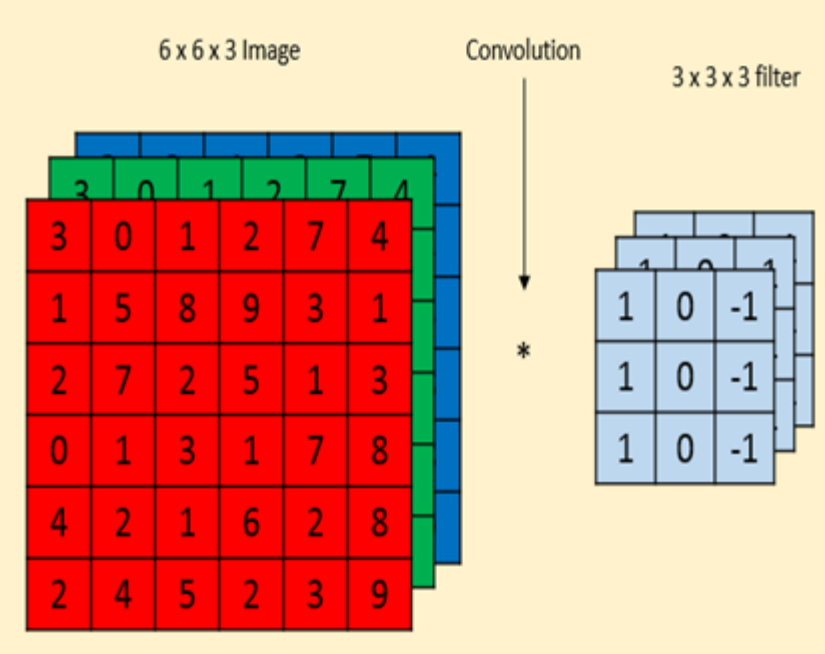
- How an image is perceived by a computer



https://engmrk.com/convolutional-neural-network-3/

Convolutional neural networks

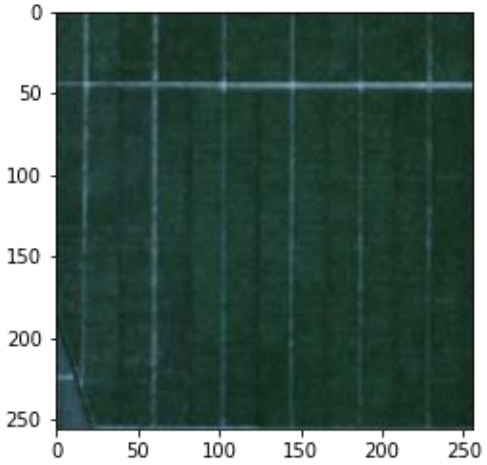
- How an image is perceived by a computer

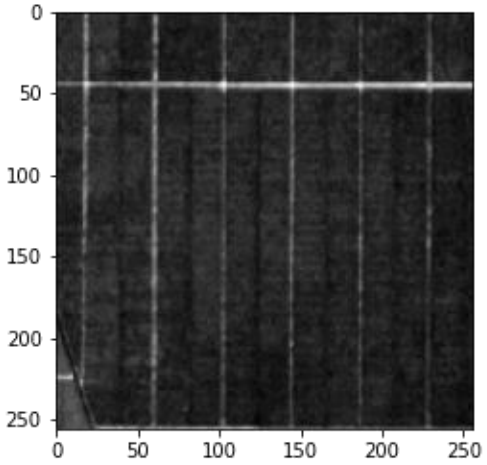


https://engmrk.com/convolutional-neural-network-3/

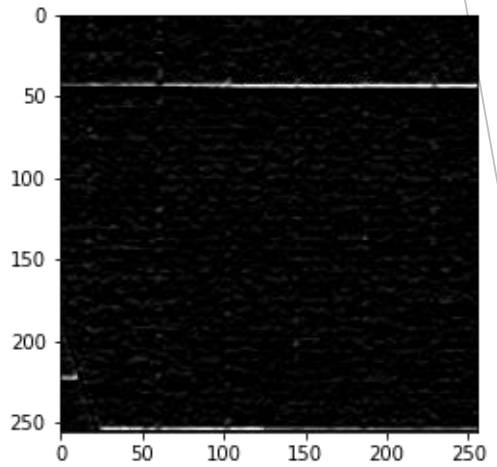
Convolutional neural networks

- Edge detection with filters

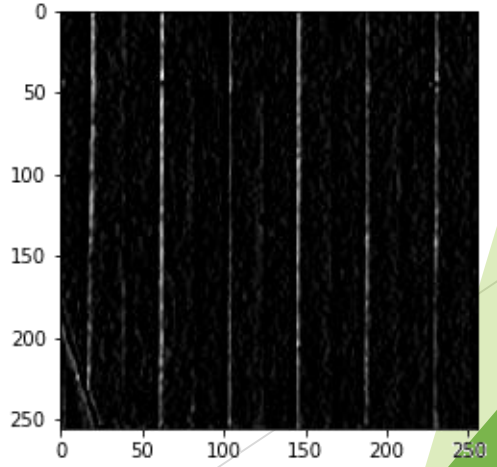




Grayscale



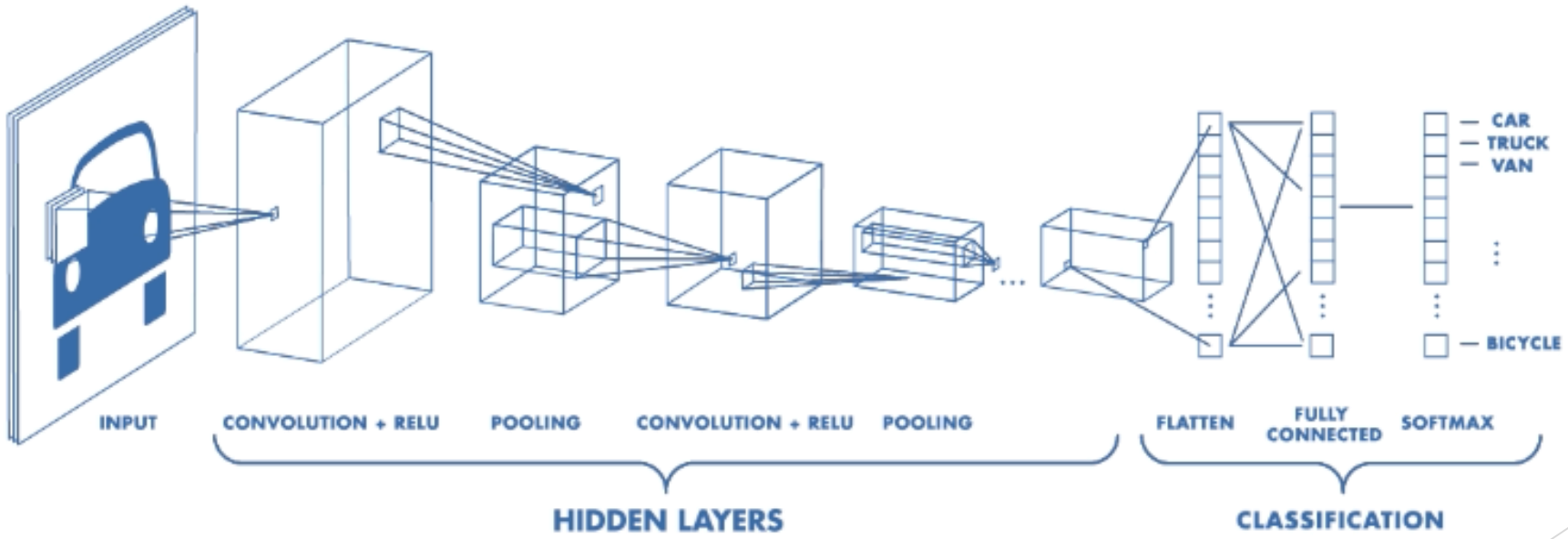
Sobel y



Sobel x

Convolutional neural networks

- Architecture



https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Class imbalance

```
# list the class sizes
print("Number of images with palm oil plantations: ", len(os.listdir(has_palm_dir)))
print("Number of images without palm oil plantations: ", len(os.listdir(no_palm_dir)))
```

```
Number of images with palm oil plantations:  936
Number of images without palm oil plantations:  14302
```

```
Class distribution of training set:
(array([0, 1], dtype=int64), array([11450,   741], dtype=int64))

Class distribution of validation set:
(array([0, 1], dtype=int64), array([1413,  110], dtype=int64))

Class distribution of test set:
(array([0, 1], dtype=int64), array([1439,   85], dtype=int64))
```

Load and transform data

```
# set data transforms
train_transform = transforms.Compose([transforms.Resize(224),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.RandomVerticalFlip(),
                                      transforms.RandomRotation([-30,30]),
                                      transforms.ToTensor(),
                                      transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
                                     ])

transform = transforms.Compose([transforms.Resize(224),
                                transforms.ToTensor(),
                                transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
                                     ])
```

Model: VGG16

- Winning model designed for the 2014 ImageNet classification challenge
- Researchers found it to be generalizable to other image sets
- Uses architecture of small 3x3 convolutional filters
- Publication: https://arxiv.org/abs/1409.1556

Model: VGG16

```
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace)
    (5): Dropout(p=0.5)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Model: VGG16

```
# freeze the features layers
for param in model.features.parameters():
    param.requires_grad = False


classifier = nn.Sequential(OrderedDict([
    ('fc1', nn.Linear(25088, 4096)),
    ('relu1', nn.ReLU()),
    ('dropout1', nn.Dropout(p=0.25)),
    ('fc2', nn.Linear(4096, 4096)),
    ('relu2', nn.ReLU()),
    ('dropout2', nn.Dropout(p=0.25)),
    ('output', nn.Linear(4096, 2))
]))

model.classifier = classifier

print(model)
```

Testing

- Confusion matrix

```
array([[1431,    8],
       [   8,   77]], dtype=int64)
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1439
1	0.91	0.91	0.91	85

Class imbalance

- To offset class imbalance, some later trainings were done on 5x duplicated minority class images

```
Number of images with palm oil plantations:  4680
Number of images without palm oil plantations:  14302
```

```
Class distribution of training set:
(array([0, 1], dtype=int64), array([11470,  3716], dtype=int64))

Class distribution of validation set:
(array([0, 1], dtype=int64), array([1441,  457], dtype=int64))

Class distribution of test set:
(array([0, 1], dtype=int64), array([1391,  507], dtype=int64))
```

Testing

- Confusion matrix of identical model hyperparameters
  - 5 epochs, SGD optimizer, 0.01 learning rate

Original dataset

```
array([[1431,    8],
       [   8,   77]], dtype=int64)
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1439
1	0.91	0.91	0.91	85

5x minority class images

```
array([[1430,   18],
       [  17,  433]], dtype=int64)
```

	precision	recall	f1-score	support
0	0.95	1.00	0.97	1435
1	1.00	0.83	0.90	463

Submission predictions

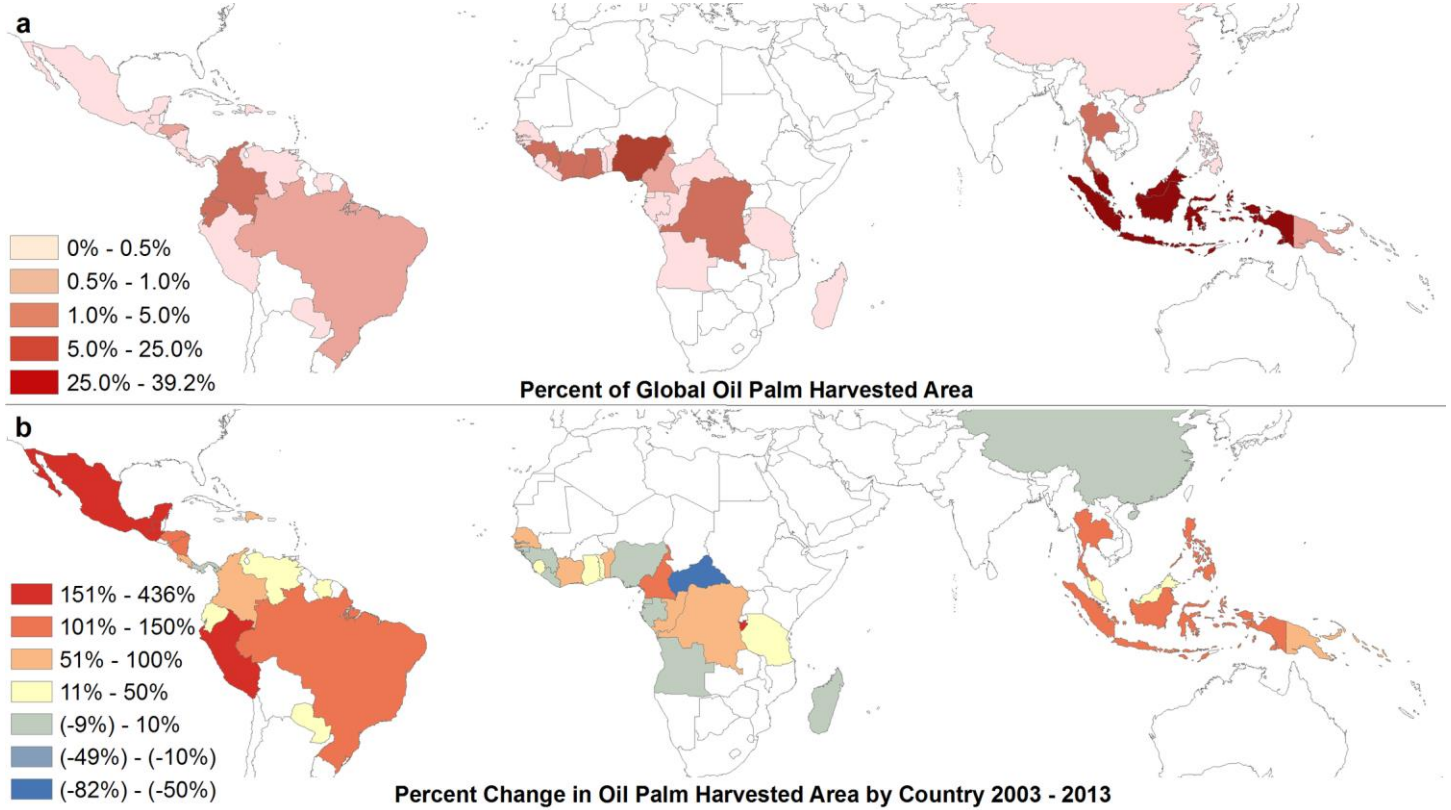
model name (.pt)	Layers + output	Dropout	Optimizer	Learn rate	Epochs	Augmentation	Testval	Leaderboard
vgg16-5ep-sgd01	3	0.5	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.99081	0.99627
vgg16-5ep-dr75-sgd01	3	0.75	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.99212	0.59369
vgg16-5ep-dr75-sgd01	3	0.75	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.98752	0.99652
vgg16-5ep-adam001	3	0.25	Adam	0.001	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.99212	0.98439
vgg16bn-5ep-sgd01	3	0.25	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.99015	0.99469
vgg16-fc4-5ep-sgd01	4	0.25	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip()	0.99146	0.99603
vgg16-imgaug1-5ep-sgd01	3	0.25	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip(), colorjitter(brightness=0.5, contrast=0.75)	0.99212	0.99634
vgg16-5ximg-5ep-sgd01	3	0.25	SGD	0.01	5	Resize(224), Randomhorizontalflip(), randomverticalflip(), randomrotation(-30,30)	0.98156	0.99207

Tweaking for improvements

- Model
  - Densenet151, Resnet121
- Adding fully connected layers
- Optimizer
  - SGD vs Adam
  - Tuning optimal learning rate (perhaps with fast.ai)
- Class imbalance
- Image processing

Q&A

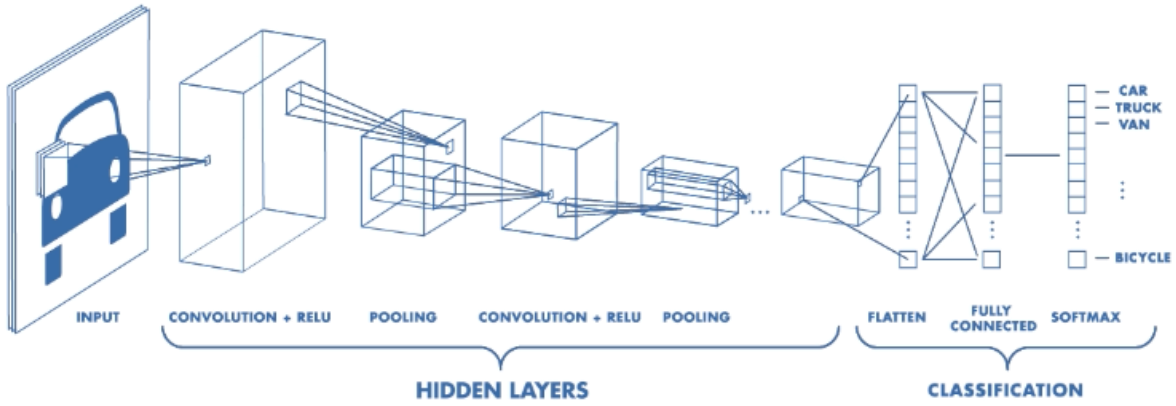
Palm oil plantation map



https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0159668

Convolutional neural networks

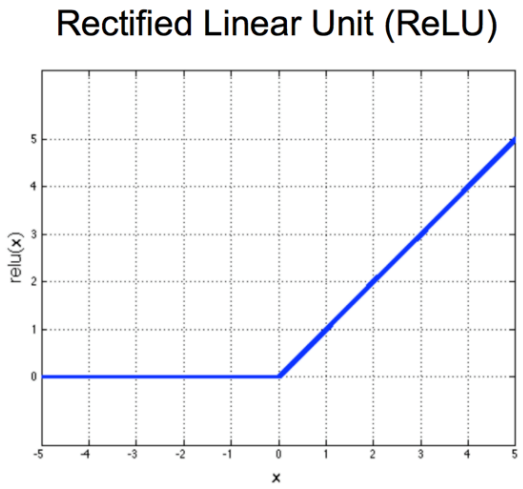
- Learning
  - Loss (mistakes between predicted and actual)
  - Backpropagation (how a particular weight is contributing to loss)
  - Optimization (how the algorithm will correct itself)



https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Activation functions

$$\text{relu}(x) \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x >= 0 \end{cases}$$



- Purpose of activation function:
  - Scales output of layers so they are consistent, small values
  - Efficiency in model training
- ReLU activation
  - Rectified Linear Unit
  - Clips negative values to zero
  - Generally should be used after every hidden layer for consistent, positive results

Image source: PyTorch Udacity Intro to Deep Learning course

Loss: Cross entropy loss

- Difference between predicted and actual classification
- First adds a softmax activation function
  - Turns scores into probabilities
- Then applies negative log likelihood loss (NLLLoss)
- Returns the average loss over an entire batch