Automatic Detection of Whale Lunges: A Deep Learning Approach

Will McCloskey
Hugo Valdivia

Roadmap

- Background + Problem Definition
- Dataset
- Results
- Algorithm Details
- Neural Networks Overview

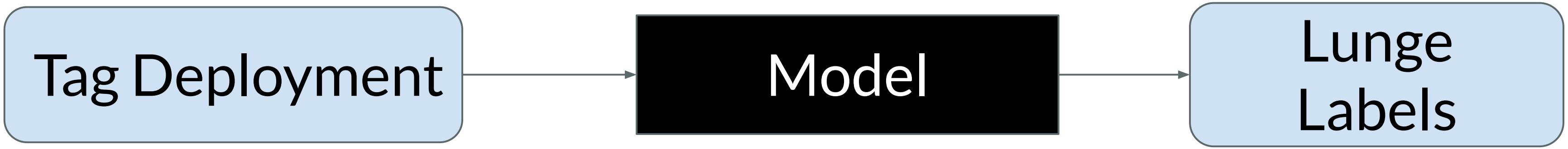
Background

- Many species of whales engage in lunge-feeding behavior
- Analysis is important for understanding the ecology, conservation, and evolution of these species
- Currently lunge feeding times are labeled by hand

We want to improve this process

Objective

Automate the labeling process!



Problem Statement

Data:

$\{x^{(t)} : t \in \{1, 2, ..., T\}\}$ Deployment time series from accelerometer measurements

$y^{(t)} \in \{0, 1\}$ Hand-labeled lunge times

Goal:

Given an unlabeled deployment, predict the lunge times.

Dataset

Species	Blue Whale	Minke Whale
Number of Deployments	29	6
Number of Lunges	4768	6151
Hours of Data	319.7	127.3
Sampling Rate	10 Hz	10 Hz

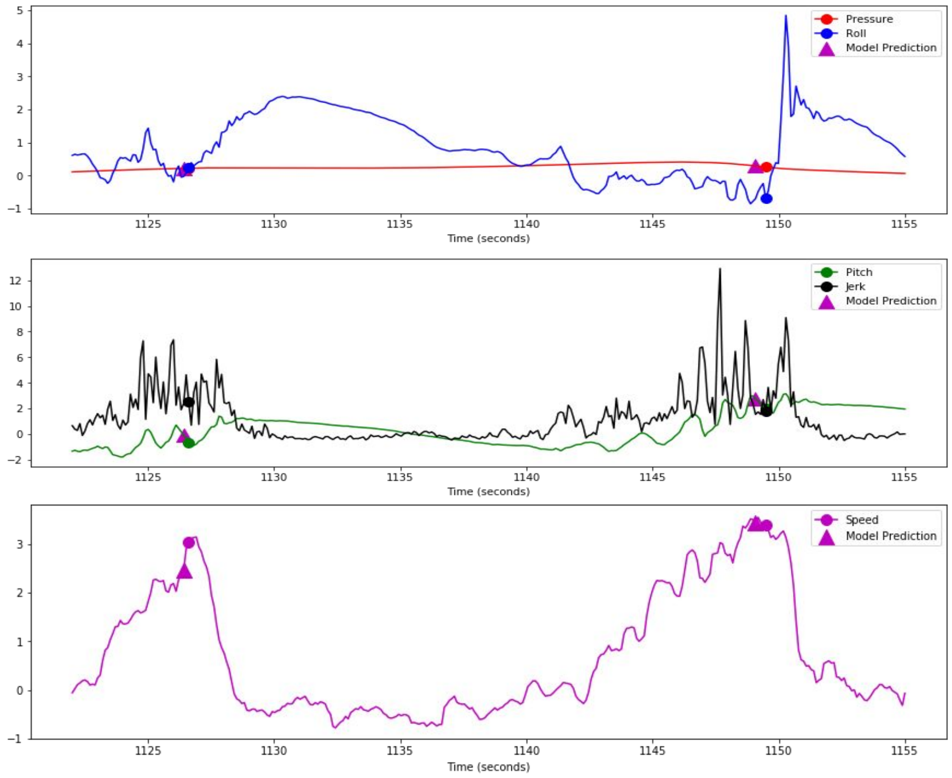
Feature Selection And Preprocessing

- Features Used
  - Speed
  - Pitch
  - Jerk
  - Roll
  - Pressure (proxy for depth)
- Preprocessing
  - Time series are normalized (mean=0, standard deviation=1) per deployment

Generalization: Key Goal of Machine Learning

- Want model to perform well on unseen data
- To achieve this, split deployments into 3 sets: training, validation, test
    - Model will see the training set but not the others.
- Training set (~80% of lunges)
    - Learning algorithm needs data to learn how to predict
- Validation set (~10% of lunges)
    - Fake test set to prevent overfitting to training set and tune accordingly
- Test set (~10% of lunges)
    - Real world performance test to assess model at the very end

Example Output (Minke Val Deployment)



Metrics

- A predicted lunge is considered correct it is within 5 seconds of a true lunge
    - True lunge = expert-labeled lunge
- True positive rate:
    - Rate at which true lunges are detected

$$\frac{\#\text{true lunges detected}}{\#\text{true lunges}}$$

- False positive rate:
    - Rate at which false predictions are made

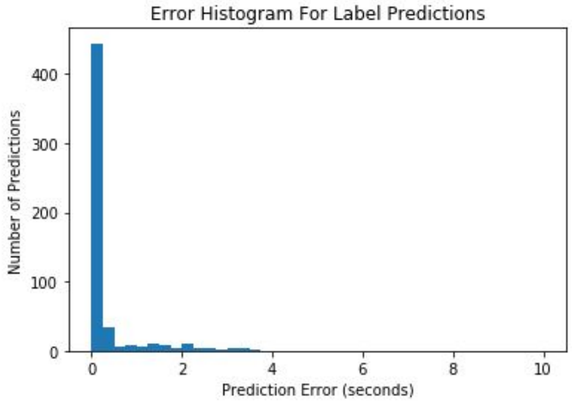
$$\frac{\#\text{incorrect predictions}}{\#\text{total number of predictions}}$$

- Number of correct predictions
- Number of predictions
- Number of overcounted correct predictions
- Average prediction error (in seconds)
- F_1 and F_2 scores

Test Set Results (Blue Whales)

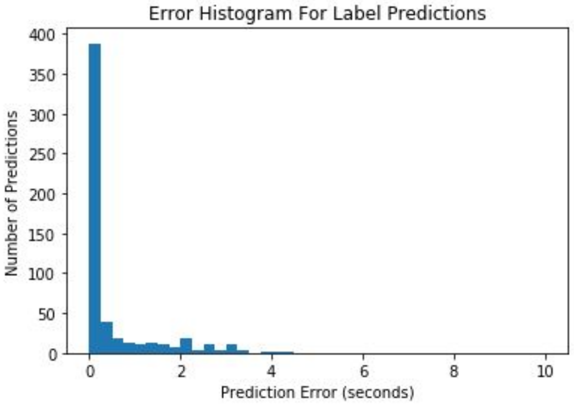
Feed Forward Model

```
The true positive rate for tolerance  5  seconds is  0.97
The false positive rate for tolerance  5  seconds is  0.037
Total lunges in files:  568
Num correct lunges:  551
Num predicted lunges:  572
We overcount by  0
We are off by an average of this many seconds:  0.31724137931034496
The f_1 score is  0.967
The f_2 score is  0.969
```



ResNet Model

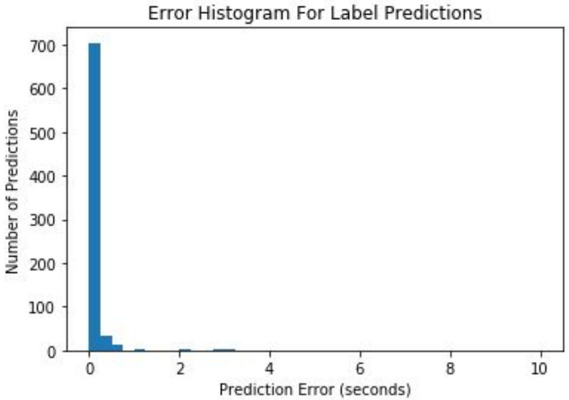
```
The true positive rate for tolerance  5  seconds is  0.972
The false positive rate for tolerance  5  seconds is  0.03
Total lunges in files:  568
Num correct lunges:  552
Num predicted lunges:  569
We overcount by  0
We are off by an average of this many seconds:  0.48550724637681136
The f_1 score is  0.971
The f_2 score is  0.971
```



Test Set Results (Minke Whales)

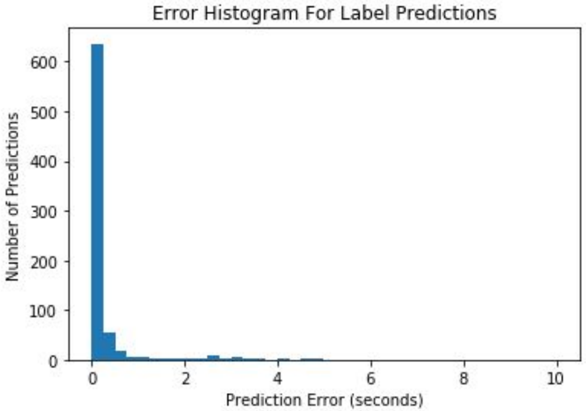
Feed Forward Model

```
The true positive rate for tolerance  5  seconds is  0.962
The false positive rate for tolerance  5  seconds is  0.088
Total lunges in files:  799
Num correct lunges:  769
Num predicted lunges:  843
We overcount by  0
We are off by an average of this many seconds:  0.14395318595578616
The f_1 score is  0.937
The f_2 score is  0.952
```



ResNet Model

```
The true positive rate for tolerance  5  seconds is  0.935
The false positive rate for tolerance  5  seconds is  0.141
Total lunges in files:  799
Num correct lunges:  747
Num predicted lunges:  870
We overcount by  3
We are off by an average of this many seconds:  0.21311914323962408
The f_1 score is  0.895
The f_2 score is  0.919
```



Clicks Saved (Test Set)

$$\#\text{clicks still needed} = \#\text{incorrect predictions} + \#\text{missed true lunges}$$

$$\#\text{clicks saved} = \#\text{true lunges - }\#\text{clicks still needed}$$

$$\text{clicks saved rate} = \frac{\#\text{clicks saved}}{\#\text{true lunges}}$$

Blue Whales (568 True Lunges)

Model	Feed Forward	ResNet
Clicks Saved	531	535
Clicks Saved Rate	93.4%	94.1%

Clicks Saved (Test Set)

$$\#\text{clicks still needed} = \#\text{incorrect predictions} + \#\text{missed true lunges}$$

$$\#\text{clicks saved} = \#\text{true lunges - } \#\text{clicks still needed}$$

$$\text{clicks saved rate} = \frac{\#\text{clicks saved}}{\#\text{true lunges}}$$

Minke Whales (799 True Lunges)

Model	Feed Forward	ResNet
Clicks Saved	695	624
Clicks Saved Rate	87.5%	78.1%

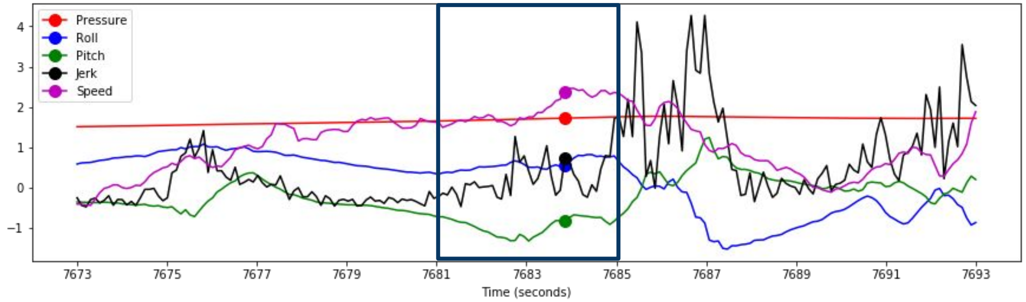
Algorithm Details

Prediction On Windows

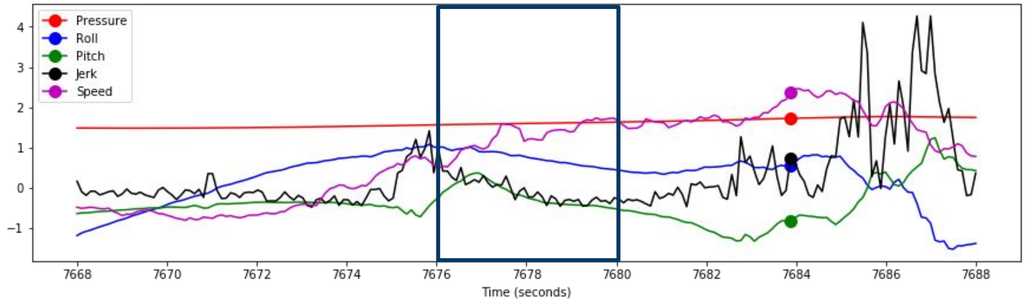
- Take a window of length T from the data
  - Assign the window a 1 if there is a lunge within x seconds of the center
  - Assign the window a 0 otherwise

Algorithm Details

Minke Whale Example (20 second windows, 2 second tolerance)



1



0

Algorithm Details

Model Prediction

- Model attempts to predict the assigned value (1 or 0)
- Model predicts probability that output is 1

Sliding Windows

- To predict at time *t*, feed the model the window centered at time *t*.
- To predict on whole deployment, make predictions every second.

Algorithm Details

Consolidating Predictions

- Output predictions can be noisy
    - Use a moving average filter to reduce false positives
- Now many predictions with probability > 0.5 correspond to a given true lunge
- To consolidate:
    - Cluster output predictions if they are within *C* seconds of each other
    - Output the max probability prediction (probability *P*) if *P*>*T* where T is a threshold value
    - Values *C* and *T* are chosen using performance on deployments in the validation set

Algorithm Details

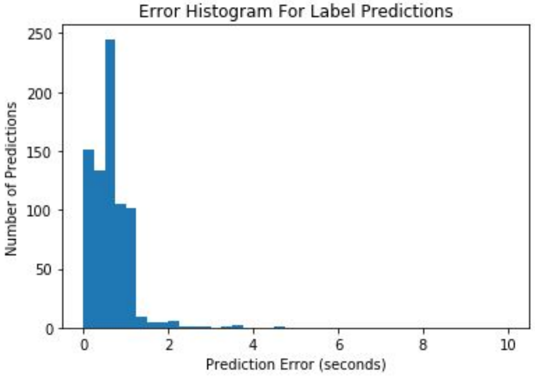
Correcting Predictions

- First (detection) model detects whether there is a lunge in the middle x seconds
  - No guarantee that predictions tightly align with true labels
- Second (correction) model
  - Takes a predicted time and predicts offset to true label
  - Trained using synthetically-generated incorrect labels
    - Offsets sampled from the distribution of errors that first model created

Correction Model Results (Minke)

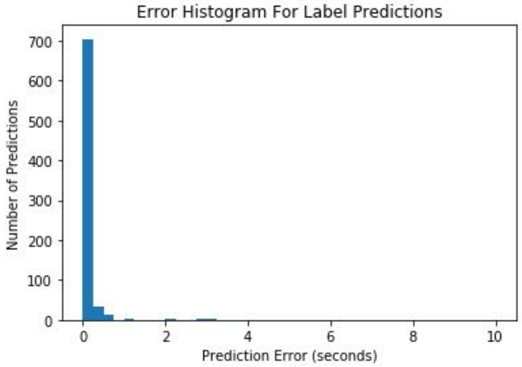
Uncorrected Predictions

```
The true positive rate for tolerance  5  seconds is  0.962
The false positive rate for tolerance  5  seconds is  0.088
Total lunges in files:  799
Num correct lunges:  769
Num predicted lunges:  843
We overcount by  0
We are off by an average of this many seconds:  0.6243172951885562
The f_1 score is  0.937
The f_2 score is  0.952
```

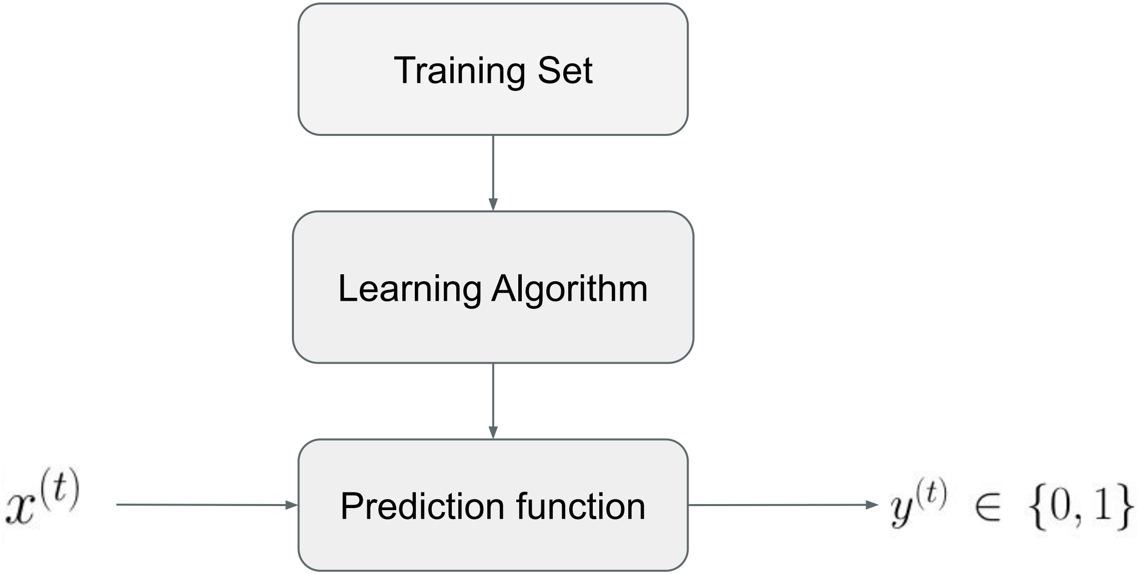


Corrected Predictions

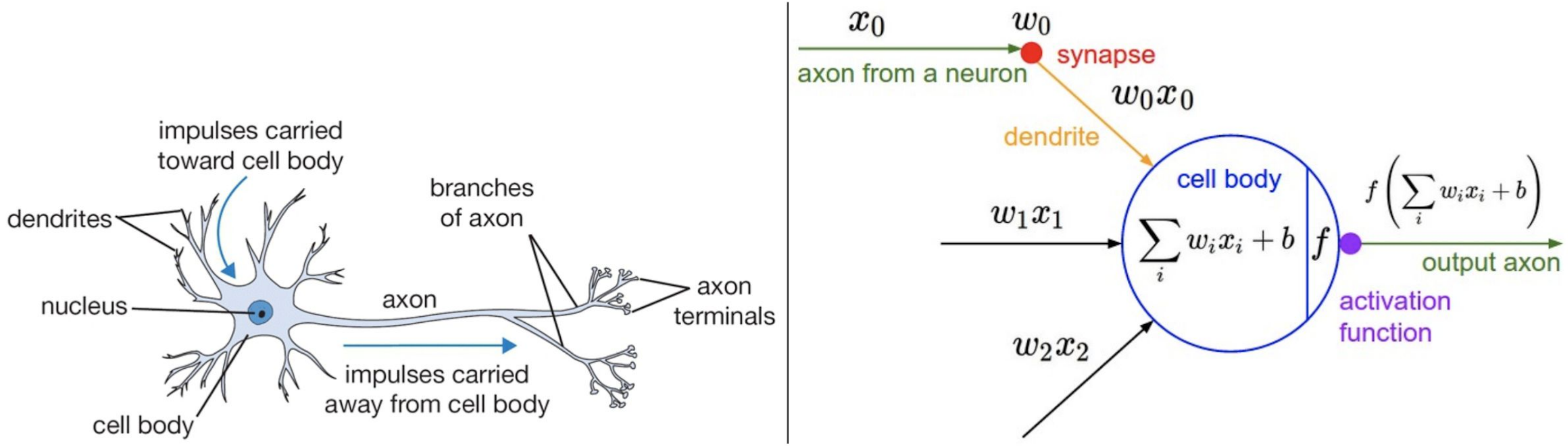
```
The true positive rate for tolerance  5  seconds is  0.962
The false positive rate for tolerance  5  seconds is  0.088
Total lunges in files:  799
Num correct lunges:  769
Num predicted lunges:  843
We overcount by  0
We are off by an average of this many seconds:  0.14395318595578616
The f_1 score is  0.937
The f_2 score is  0.952
```



Supervised Learning: Binary Classification



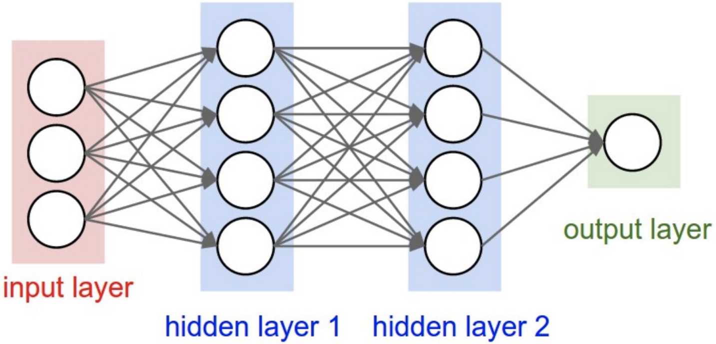
Neural Networks



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Source: CS231N Lecture Notes (http://cs231n.github.io/neural-networks-1/)

Neural Networks

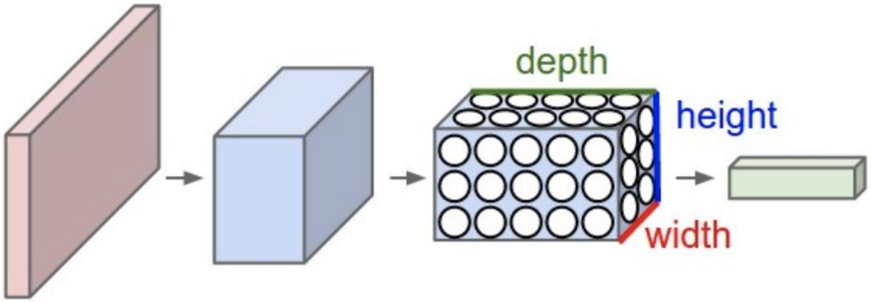


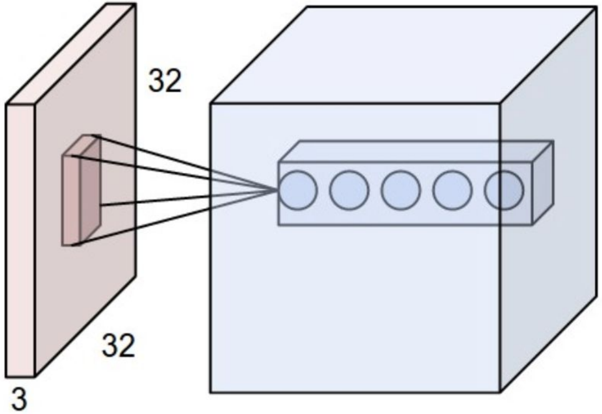
- Connected, acyclic graph of neurons
- Single neuron linear classifiers
  - Can be equivalent to logistic regression and SVM
- General feed-forward neural networks
  - A series of matrix multiplications linked with non-linearities
- Representational Power
  - Single hidden layer is enough to approximate any continuous function
  - However, just because a function can be learned doesn't mean it will be learned
    - Empirically, deeper networks will work much better in practice but must be tuned appropriately
  - Key takeaway: neural networks are able to learn well-behaving functions that fit real-world data tremendously well.

Source: CS231N Lecture Notes (http://cs231n.github.io/neural-networks-1/)

Convolutional Networks

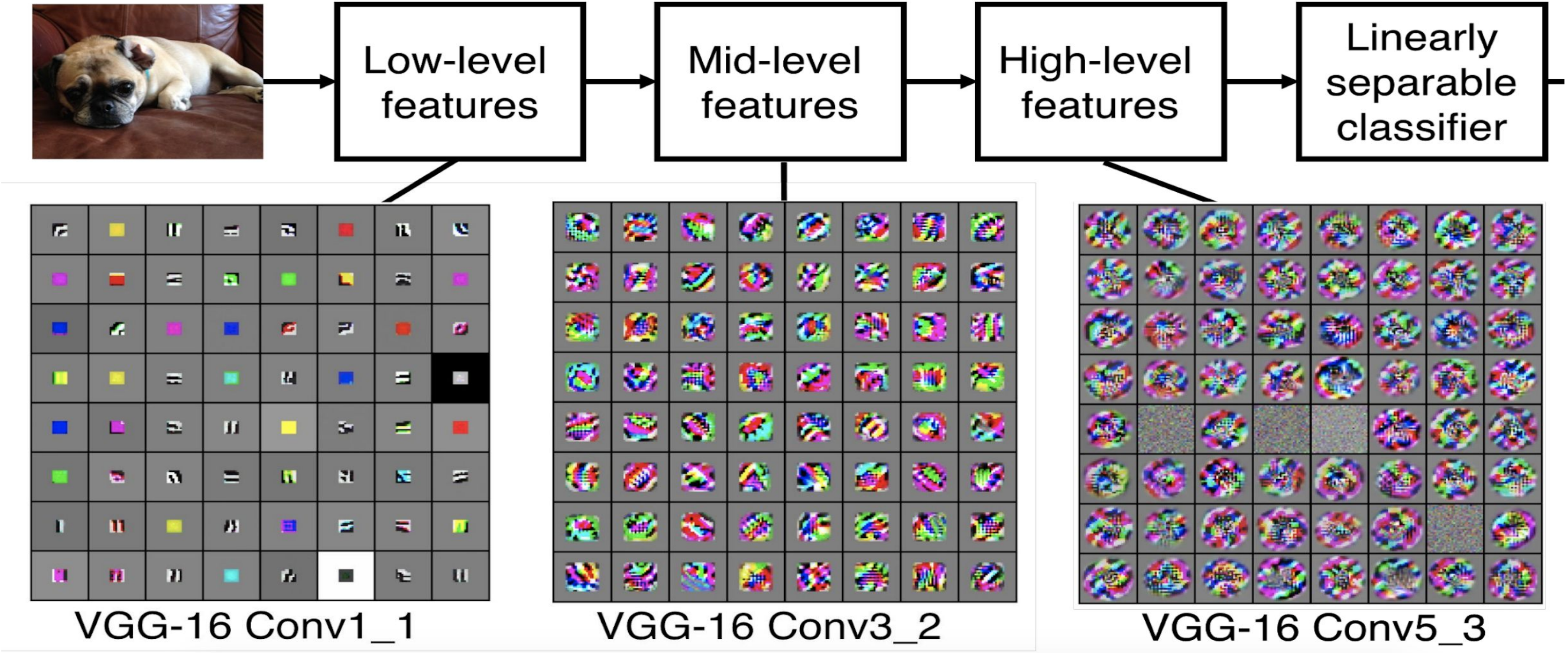
- Optimized architecture for images
    - Parameter sharing to decrease number of parameters
- Typical architecture consists of CONV, POOL, and FC layers
- Most important is CONV
    - Small filters slide across image to output a new image that holds the filter's response to each spatial section
    - Filters will activate when they detect a certain feature





Source: CS231N Lecture Notes (http://cs231n.github.io/convolutional-networks/)

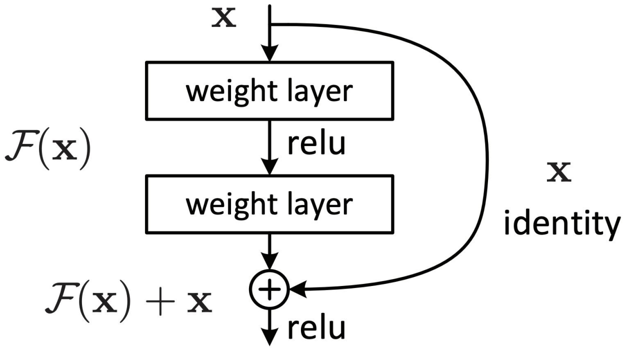
Convolutional Networks: Visualized



Source: CS231N Lecture Notes (http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf)

Residual Networks (ResNets)

- Trend towards deeper networks
  - Makes sense to learn more low/mid/high-level features
  - But much harder to train
- ResNets allow for much deeper networks (~1000-layer networks)
- Have been found to work quite well on time series data



A residual block

Source: He, Kaiming et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): n. pag. Crossref. Web.

External Resources

- Code base can be found at:
  - https://github.com/valdivia4/Deep-Learning-Lunge-Detection
- Implemented in Python
- Detailed documentation provided
  - No prior Python experience necessary