AniML

No Code, ML Vision for Specialists with no CV experience

Presented by:
Ivan Wong ,
Lana Elauria,
Lucas Harvey-Schroyer,
Whit Blodgett

UC Berkeley MIDS, W210 Fall 2022

Agenda

Problem & Mission Statement

Impact

MVP overview

Technical Discussion

Evaluation

Recommendations & Next Steps, Acknowledgments, and Conclusion

Problem being solved

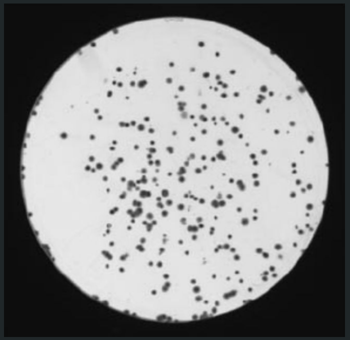
As a Specialist with no computer vision experience, I need a way to rapidly create a CV system to (1) automatically filter a large set of images, (2) identify objects of interest and (3) provide analytical insights.

Mission Statement

Empower specialists to spend less time manually reviewing images and more time applying their unique skills, by democratizing access to computer vision.

Impact

Biologist → Species
20m globally



Life Scientist → Cells
$230bn Life Science Tool Market Cap

QA Manufacturer → Defects
Poor quality degrades 15-20% total manufacturing revenue ($5bn lost in phones in 2017)

Demonstration of the MVP

Website accessible by end user, including:

- *Image Filtering for ‘false triggers’*
- *Re-training of the Yolo V5 model based on user’s trained images*
- *Image detection*
- *Summary analytics of predicted classes*

The no-code, few shot data, user friendly platform hopes to demystify computer vision applications for our target user(s).

Home Page	Filtering	Labeling & Training	Inference	Results & Analytics
Home page describes the website’s layout and highlights major modeling steps	Filtering allows the user to upload a large volume of images do automatically filter ‘false triggers’	The user then can label around 20 images to retrain the Yolo V5 model with additional classes	With the newly trained model, the user can upload they filtered images to automatically classify objects of interest	Finally, the user can adjust confidence thresholds per class and view histograms of model predictions to adjust model classifications and save results

User Interface and Experience

Initial User Interview (10/7)

MVP User Test (12/5)

Erin the biologist

It takes so long to manually sort through 250,000 images!

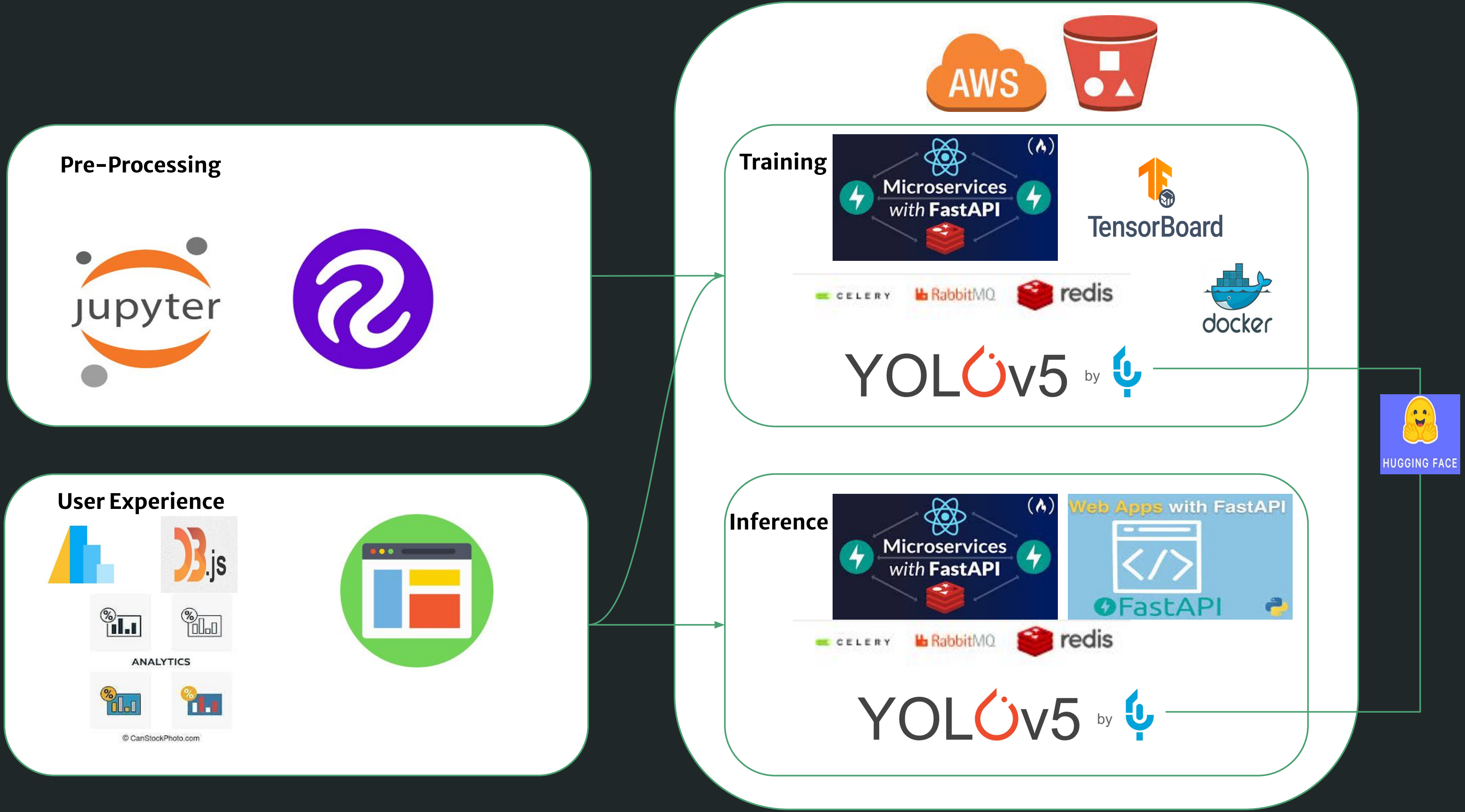
The current state-of-the-art is so confusing, and it still takes hours to run!

“You’ve saved me at least five hours a week!”

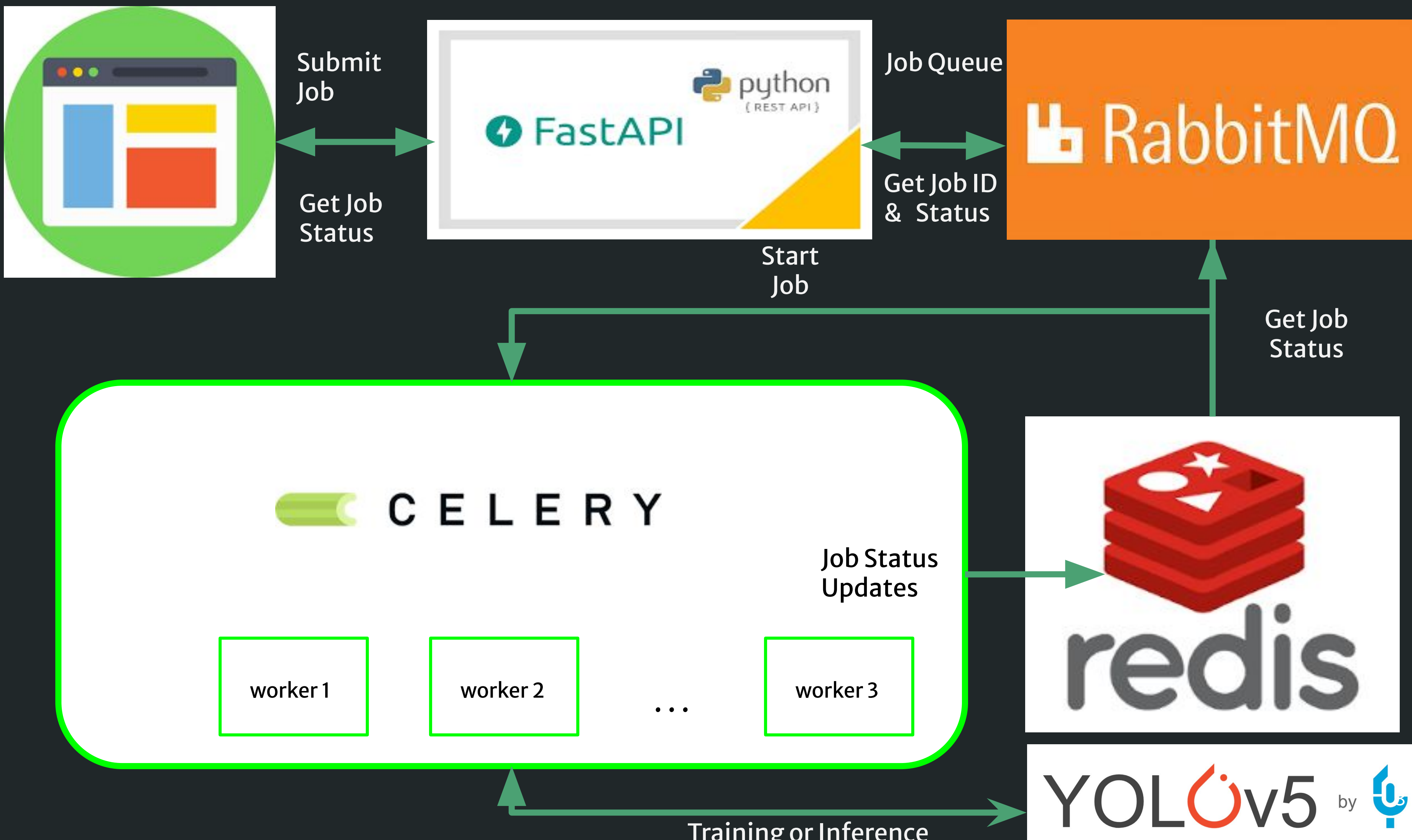
“The less time we spend processing our data, the more time we can spend turning our findings into manageable insights for wildlife management agencies.”

“It’s very ‘coding illiterate’ friendly!”

Product Tech Stack



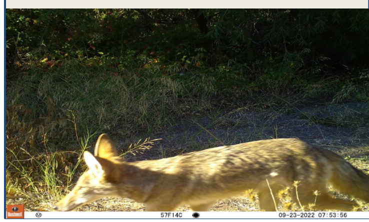
Async Architecture for Large Dataset



Data Pipeline

Filtering

Input Data: Images

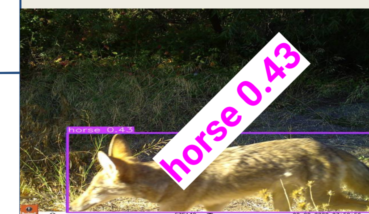


Pre-processing

Scaling down images and removing "No Object of Interest"

YOLOv5x trained on 80 classes

Weighted Boxes images



Detection

Input Data: Images

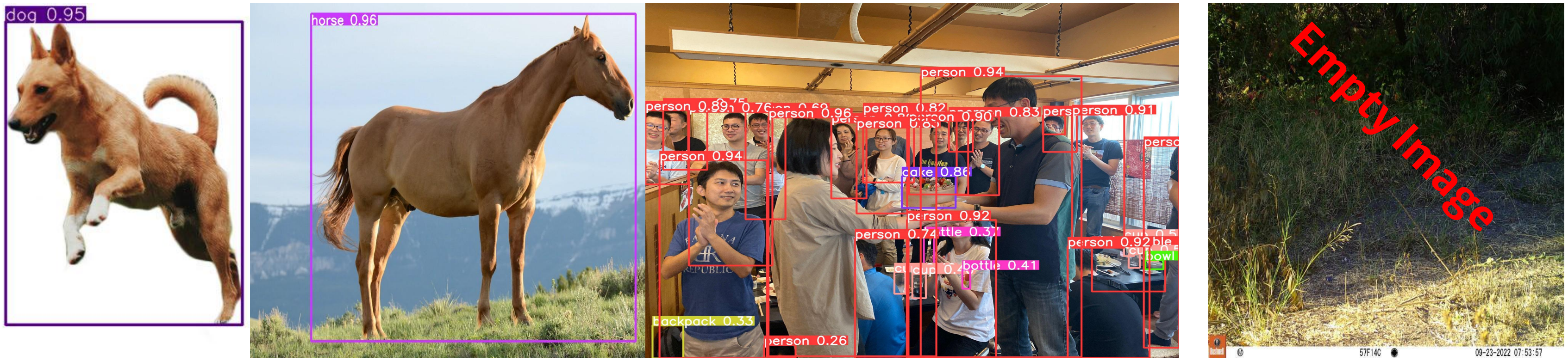
Pre-processing

Scaling down images, removing "No Object Interest", and passing with new classes

YOLOv5x custom trained on new classes

Input: Annotated Images

Model Insights and Outputs



Input: 640x640 image

Output:
[cx, cy, w, h, conf, pred_cls(80)]

Output Example:

"x": "0.6173712",
"y": "0.6719394",
"w": "0.2972912",
"h": "0.20354787",
"prob": "0.8142258",
"class": "Fox"

Challenges and Tradeoffs

Main takeaway: model choice and data needs are highly dependent on a user's specific CV application

Technical Challenges	Data	Ability to handle large dataset (i.e over 100, 000 images)
	EDA	Determine the confidence level for filtering
		Minimize false-negative from blurry/partial images
		Evaluate "what is few-shot training" for our use case
Performance Trade-offs	User Needs	Filter out null images or capture partial images, i.e. decrease precision and F1 scores to increase recall
	Model Choice	Detectron2 is more accurate while YOLOv5 is faster and more efficient at prediction
		YOLOv5 training time is double compared to Detectron2. Training Data Size has lower accuracy, mAP, with small training data size, i.e < 150 images.

Model Choice:
model size

14.5 mb

YOLOv5

Model Choice:
performance pros & cons

Model Name	Params (Million)	Accuracy (mAP 0.5)	CPU Time (ms)	GPU Time (ms)
YOLOv5n	1.9	45.7	45	6.3
YOLOv5s	7.2	56.8	98	6.4
YOLOv5m	21.2	64.1	224	8.2
YOLOv5l	46.5	67.3	430	10.1
YOLOv5x	86.7	68.9	766	12.1

User needs:
capture partials vs filter empty images

ML Models Comparison

	YOLOv5	MASK R-CNN	Detectron2
Inference Speed	●	◐	◐
Detection of small or far away objects	●	◐	◐
Little to no overlapping boxes	●	◐	◐
Missed Objects	◕	◕	◐
Detection of Crowded objects	●	●	◐
Smaller model size	●	◐	◐
Training data size < 150 images	◕	◐	●
Training Resource Usage	●	◐	◐
Training Time	◕	◐	●
Accuracy	◕	◐	●

Main takeaway: for our use case of animal detection and classification, YOLOv5 was the clear winner

Technical Key Takeaways - CV Application

Model: Convolutional Neural Network (CNN) is best choice for image processing and CV applications.

- Training and Detection: We used Ultralytics **YOLOv5** APIs.

How we **bootstrapped** our CV application development (see Appendix for more details).

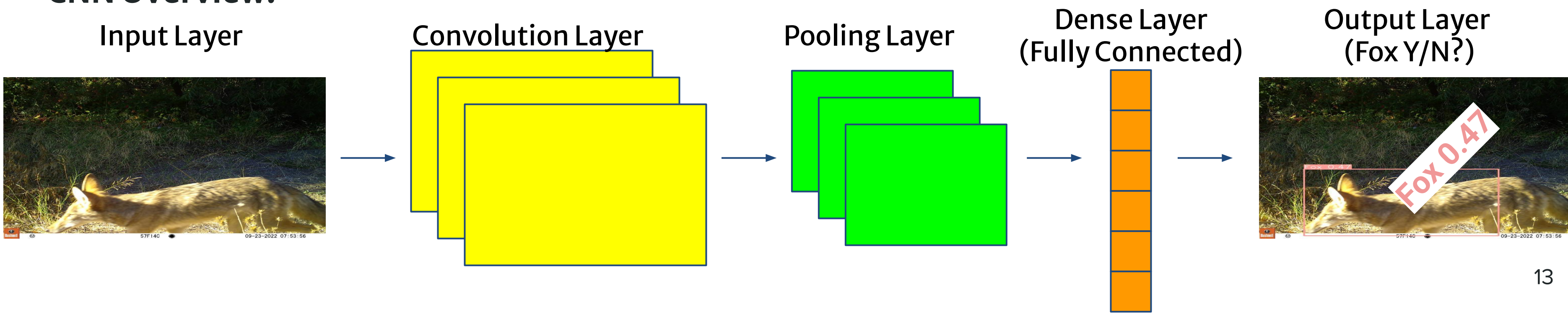
- Amazon for Infrastructure & Development Tools (e.g. notebooks):

Amazon tools for infrastructure and development

- Other tools for the Framework & Optimization:

Fast API, Celery, RabbitMQ, and Redis for the front end & backend application framework and data storage

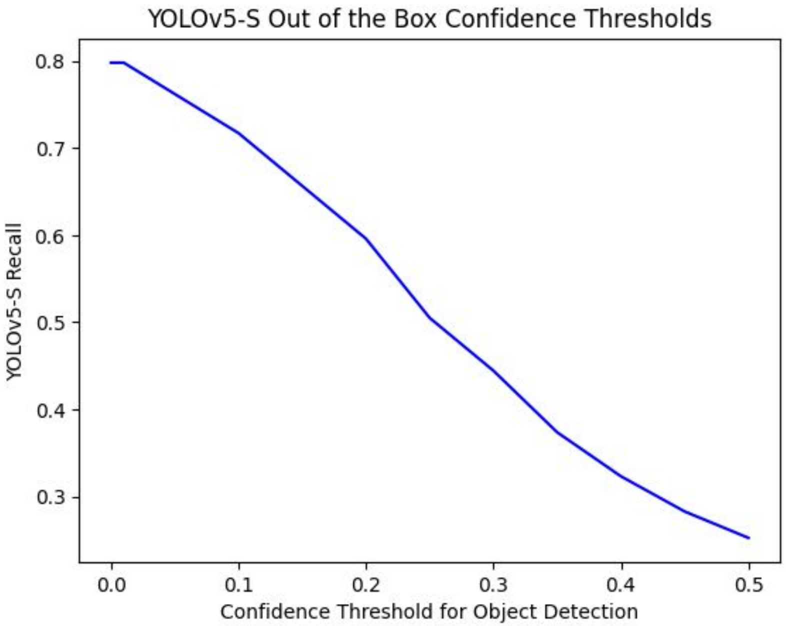
CNN Overview:

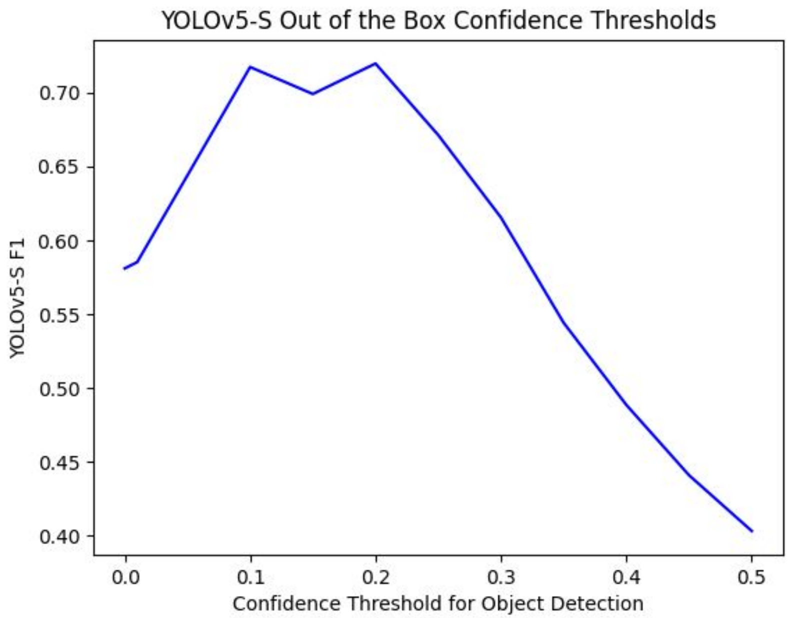


Technical Evaluation

Phase 1: Filtering empty images with the standard YOLOv5 (i.e. no training with our data)

- *The main goal was to maximize recall while still filtering out as many null images as possible*
- *Arrived at 0.1-0.2 ideal confidence threshold to filter out nulls while still capturing most images of interest*
    - *This low baseline threshold illuminated the need for custom training capabilities*

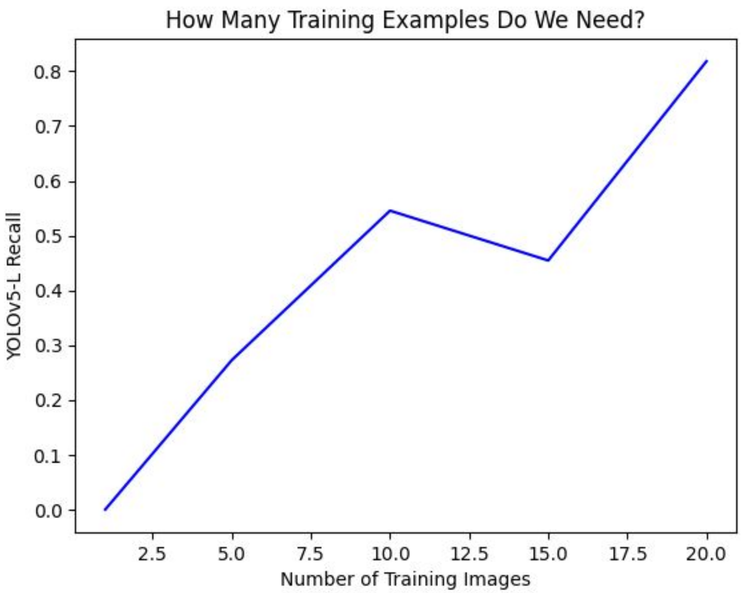


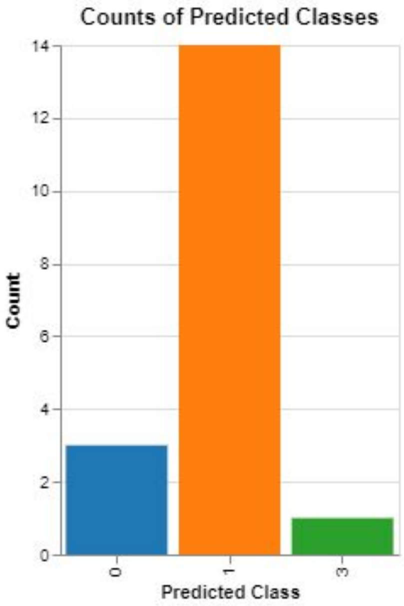


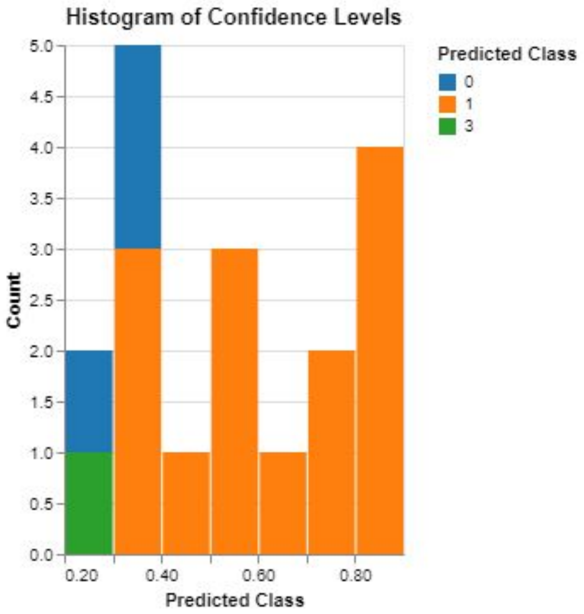
Technical Evaluation (contin.)

Phase 2: Classifying with custom-trained YOLOv5 on our user's data

- *Our main goal was to determine how many training images we needed the user to provide*
- *Also wanted to create a framework that visualized model results in a way that a non-technical user would easily understand, empowering them to evaluate their own model at runtime*

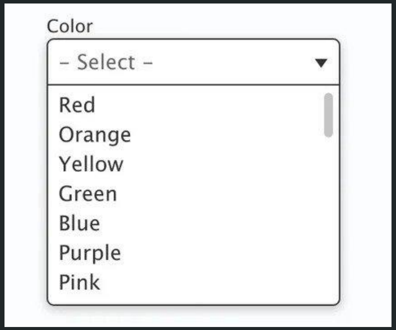






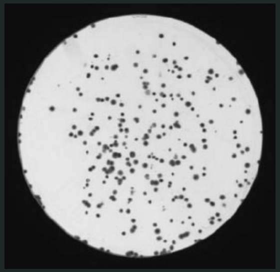
Recommendations & Next Steps

- *Polishing analytics visualizations and increasing flexibility for users*



- *Adding a self-contained annotation workflow*

- *Expanding use case testing*



- *Automating customized model deployment to scale AniML's user base*

Wrap Up

As a Specialist with no computer vision experience, I need a way to rapidly create a CV system to (1) automatically filter a large set of images, (2) identify objects of interest and (3) provide analytical insights.

AniML reduces the amount of time needed for non-technical tasks and enables these groups to spend more time *where it counts.*

ML Vision Models →	*Reduce time for non-technical tasks* by orders of magnitude AND can automatically classify species of interest
Web Interface →	*Easy to understand*, no code solution enables user to automatically process the data
Data Centricity →	*Few shot model* approach ensures the app adds value and doesn't increase total time needed compared to manually processing data

Acknowledgements

Team Responsibilities

- *Ivan Wong: Infrastructure & data engineer, machine learning engineer*
- *Lana Elauria: Machine learning engineer, business case development*
- *Lucas Harvey-Schroyer: Project manager, UI/UX developer*
- *Whit Blodgett: Product manager, business case development*

We would like to acknowledge and thank Erin Weiner and her wildlife research team at CSU Long Beach for allowing us to use their data and providing us with valuable user feedback throughout the project.

Also a huge thank you to Joyce Shen and David Steier for facilitating our capstone and providing us with feedback on all aspects of our project!

Q + A

Contact:

Ivan Wong: ivanwong@berkeley.edu

Lana Elauria: lana.elauria@berkeley.edu

Lucas Harvey-Schroyer: lschroyer64@berkeley.edu

Whit Blodgett: wwblodge@berkeley.edu

Appendix

Technical Key Takeaways - CV Application

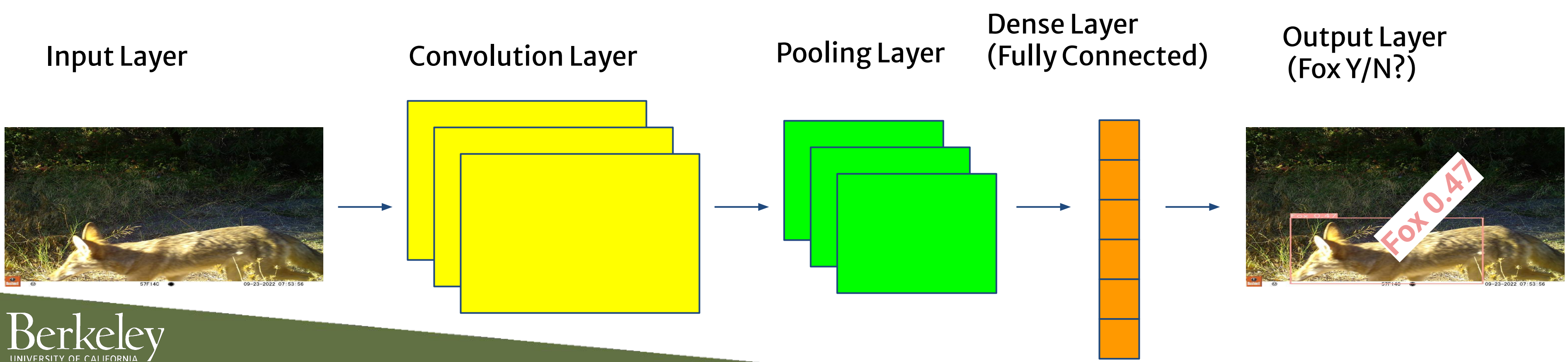
YOLOv5 by

- Model: Convolutional Neural Network (CNN) is best choice for image processing and CV applications.
- Training and Detection: Ultralytics YOLOv5 APIs.
- **Bootstrapping** a CV application development (Infrastructure):
  - AWS EC2 Instance:
    - Machine with GPU g4dn.xlarge
    - Application and OS Images (AMI): NVIDIA GPU-Optimized AMI v22.06.0
  - Notebook: Sagemaker Studio Lab
  - Framework: FastAPI frame for the microservices and the web server.
  - Optimization: Celery, RabbitMQ, and Redis for async architecture.

Amazon EC2

amazon SageMaker Studio Lab

CELERY RabbitMQ redis

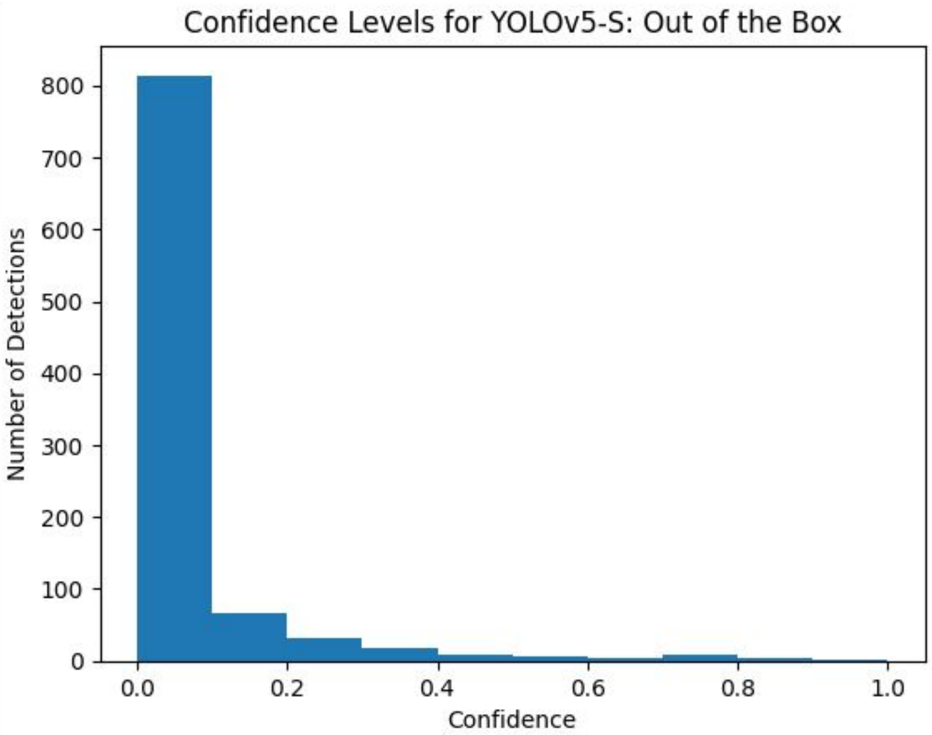


Berkeley
UNIVERSITY OF CALIFORNIA

Data and EDA

EDA:

It is crucial that we balance the number of false positives that we filter out with the number of true positives that may be predicted with low confidence.

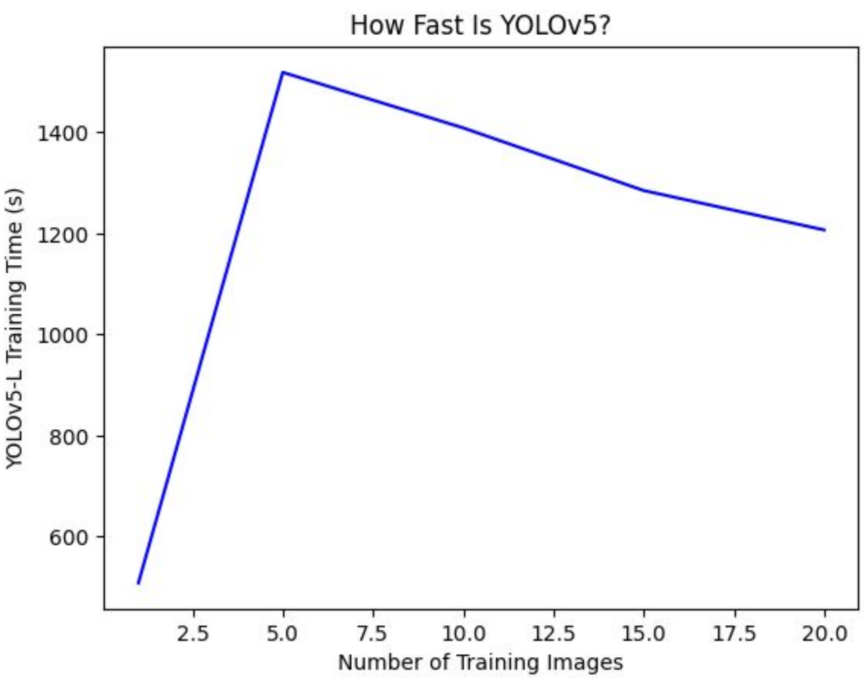


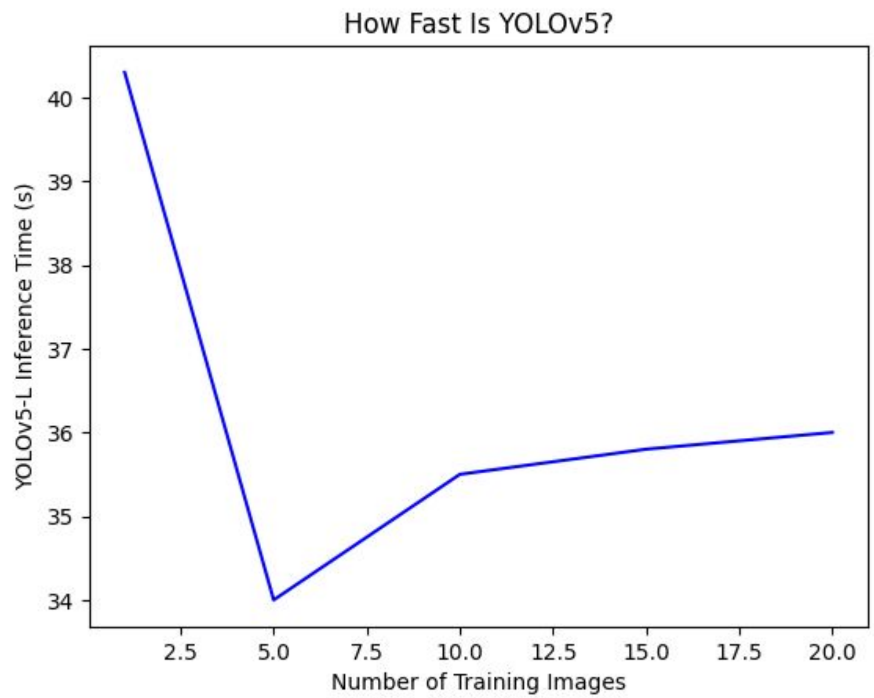
Technical discussion

Model Evaluation:

Training time mostly consistent around 20 minutes

Inference time consistent around 35-40 seconds (as opposed to 22 minutes

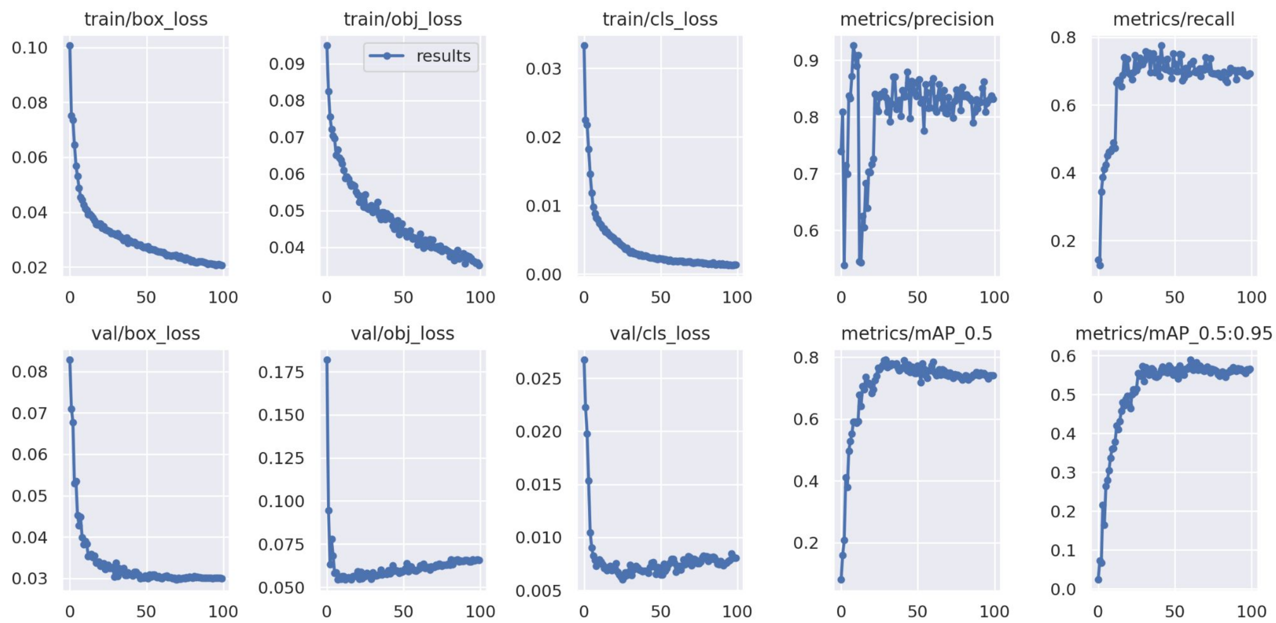




Challenges & Tradeoffs

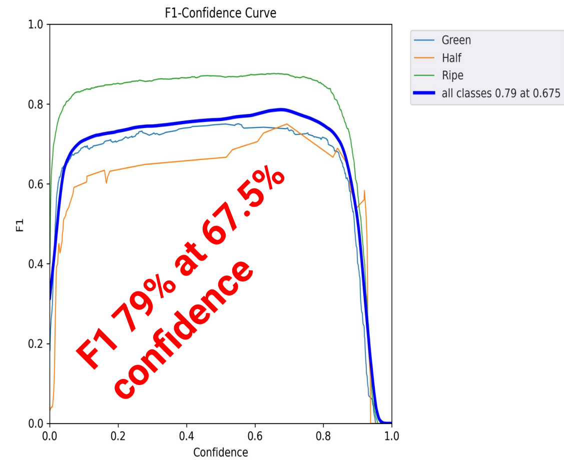
- Training Data Size vs Accuracy: < 150 images, low mAP (mean Average Precision)
- Training time is double compared Detectron2. YOLOv5 offers various size that meet the application need.

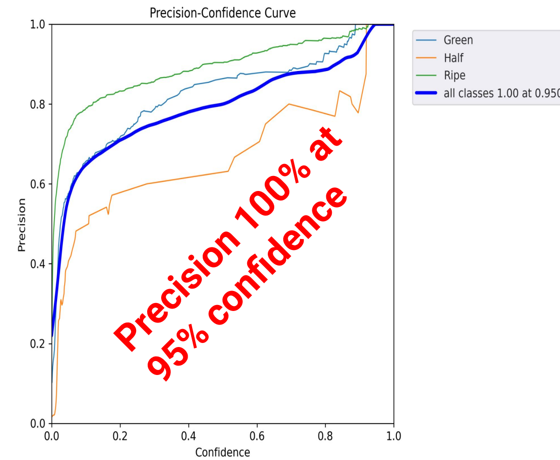
Model Name	Params (Million)	Accuracy (mAP 0.5)	CPU Time (ms)	GPU Time (ms)
YOLOv5n	1.9	45.7	45	6.3
YOLOv5s	7.2	56.8	98	6.4
YOLOv5m	21.2	64.1	224	8.2
YOLOv5l	46.5	67.3	430	10.1
YOLOv5x	86.7	68.9	766	12.1



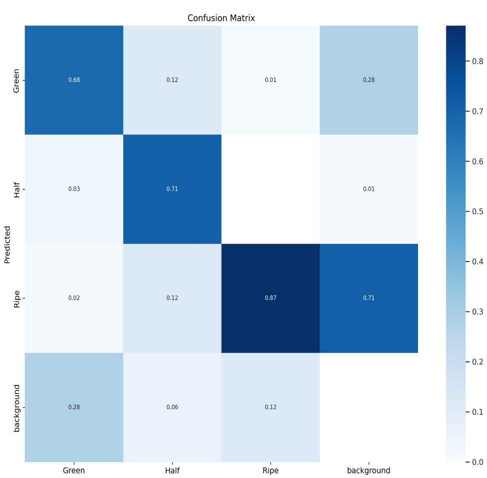
Training Results

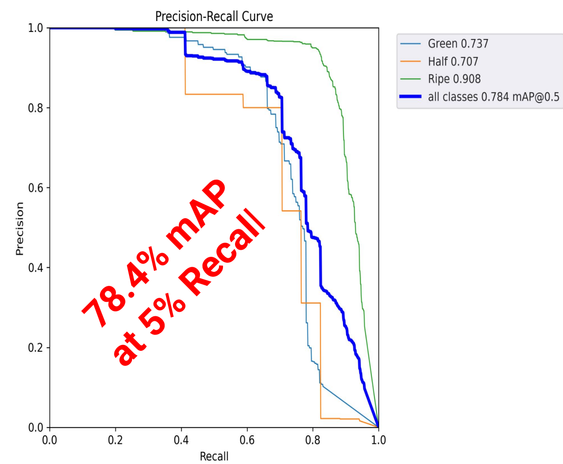
YOLOv5 by

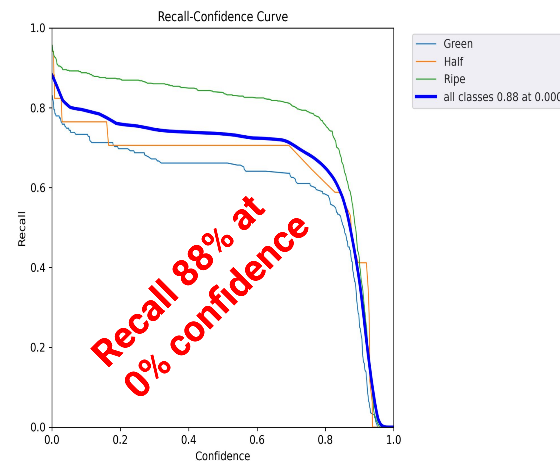




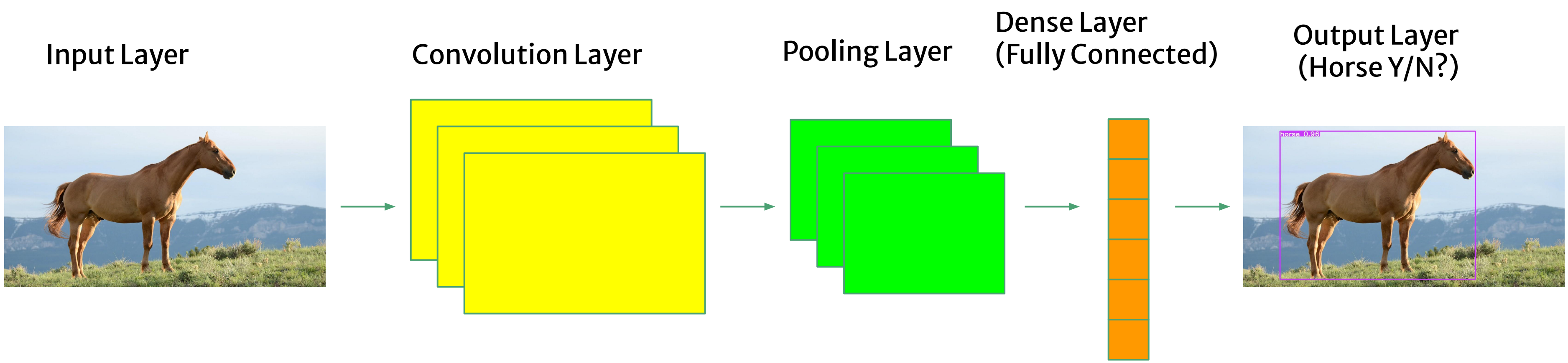
Val batch with prediction and Label







Convolutional Neural Network (CNN)



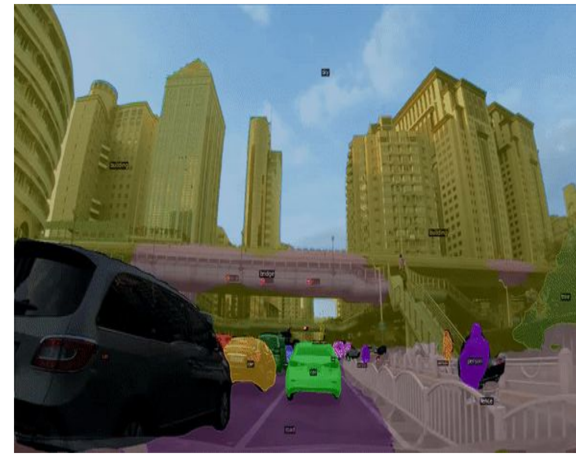
- **Network architecture for deep-learning algorithms.** Identifying and recognizing objects..
- **Identify patterns.** Pixel data process, Leverages principles from linear algebra, such as matrix multiplication to
- **Computer vision (CV) applications.** Self-driving cars and facial recognition.
- **CNN Layers.** The CNN **learns the object's features in successive iterations as the object data moves through the CNN's many layers.**
  - **A CNN can have multiple layers. Minimum 3 layers.** Repeated operations for dozens, hundreds or even thousands of layers.
  - A **filter or kernel is applied to each image to produce an output** and passes on its output to the filter in the next layer.
  - **Identify the entire object.** All the **image data progressing through the CNN's multiple layers.**
- **Transfer learning.** Retrained for new recognition tasks and built on pre-existing networks.
- Lower computational complexities or costs.
- Other Modeling techniques:
  - One type of an **Artificial neural networks (ANNs)** is a **recurrent neural network (RNN)** that uses sequential or time series data as input. It is suitable for applications involving **NLP, language translation, speech recognition and image captioning.**

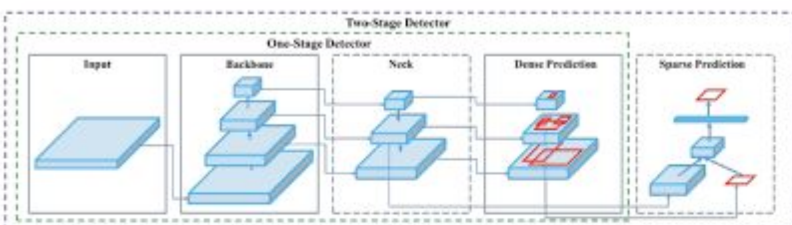
ML Models

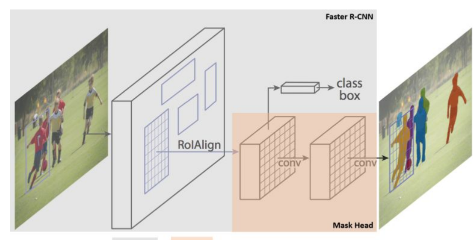
YOLOv5 by

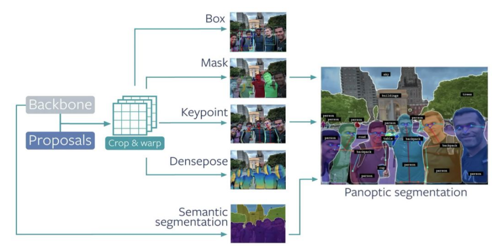
MASK R-CNN
OBJECT SEGMENTATION

Detectron2









2nd-Stage: Sparse Prediction.

Backbone: CSP Extraction of informative features.

Neck: PANet. Elaborate in feature pyramids.

Head: Yolo Layer. Bounding box, class, score.

Faster R-CNN + Mask Head → Mask R-CNN
Mask R-CNN developed on top of Faster R-CNN.
Faster R-CNN is a region-based convolutional neural networks for Instance segmentation.

Detectron2 is a robust version of Mask R-CNN architecture.
PyTorch framework.
New functionality. Densepose, Cascade R-CNN, rotated bounding boxes, panoptic segmentation, etc.
More modular design and flexibility to train at high speed on single or multiple GPU servers.