CS230

COVID-19 Vaccination Card Classification and Features Detection

Yingxiao Liu
Department of Civil and
Environmental Engineering
liuyx@stanford.edu

Xiaojuan Liu
Department of Epidemiology and
Population Health
xjliu@stanford.edu

1 Introduction

A COVID-19 Vaccination Record card is designed by the Disease Control and Prevention (CDC) and issued to people vaccinated in the United States as a personal vaccination record. A complete vaccination card shows what COVID-19 vaccine a recipient received, as well as the date and where the recipient received it. While it's not a legal document, more and more policies require it as a proof of vaccination. The CDC recommends recipients to not only keep the card for future use but also save a picture of the card as a backup copy.

As businesses begin to reopen, people need proof of vaccination to work in the federal government, at tech firms such as Google, Facebook, and a growing list of other companies. Organizations and companies may also require their employees and customers to upload a picture of the card to their online system so as to get full access to the facilities and in-person activities. However, the pictures uploaded may not always be valid. First, people may not upload a "real" card picture (but an image showing a close or even random pattern) trying to fool the system. Second, people may take blurred pictures of the card with their cell phones, or the card in the image may be incomplete/truncated or with missing/blocked fields (e.g., a CDC logo). This gives rise to the need for an approach to efficiently classify and verify the images of vaccination cards.

In this project, we aim to use deep learning methods to identify vaccination cards by image classification and locate the key features of the card by object detection given a card is present in an image. Fig. 1 shows a flowchart of works performed in this study. To the best of our knowledge, this is the first work classifying vaccination card and using object detection algorithm to detect vaccination card features. Automatic detection and recognition of multiple fields/features from the vaccination card can replace manual key-in and significantly improve the workflow in the real-world usage scenario.

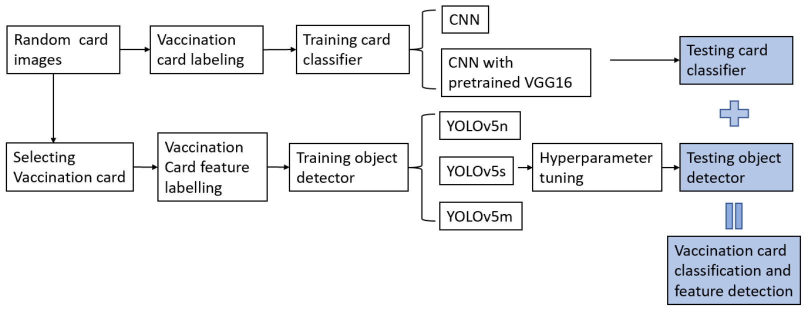
Random card images
Vaccination card labeling
Training card classifier
CNN
CNN with pretrained VGG16
Testing card classifier
YOLOv5n
Selecting Vaccination card
Vaccination Card feature labelling
Training object detector
YOLOv5s
Hyperparameter tuning
Testing object detector
YOLOv5m
Vaccination card classification and feature detection

Figure 1: Pipeline of works performed in this project

2 Related Work

Object detection, a major focus of this project, has been a rapid revolutionary field in deep learning, and is closely related to many applications, including image classification [1, 2], human behavior analysis [3], face recognition [4] and autonomous driving [5, 6]. Since the proposal of Regions with CNN features (R-CNN) [7], a great number of advanced algorithms for object detection have been established: including Fast R-CNN which jointly optimizes classification and bounding box regression tasks [8], Faster R-CNN which takes an additional sub-network to generate region proposals [9], R-FCN which is fully convolutional with all computation shared on the entire image [10] and YOLO which accomplishes object detection via a fixed-grid regression [11]. Previous works have adopted YOLO algorithm to detect different fields of identity documents such as the driver's license [12, 13]. However, prior works assumed that all images were valid and all fields of interest were present in the image, while in our project, we considered the images of both fully valid vaccination cards and invalid cards.

3 Dataset

Data was primarily collected online. A total of 388 images of three types were collected:

(a) images showing a valid CDC COVID-19 vaccination card with 3 complete key features (header, CDC logo and vaccine history);

(b) images showing a vaccination card but some key features are truncated or missed;

(c) images showing other types of cards or information, including card/record of other vaccine, vaccine card of other countries, debit card, credit card, student card, name card, driver license, passport, postcard, gift card, travel tickets, and receipts.

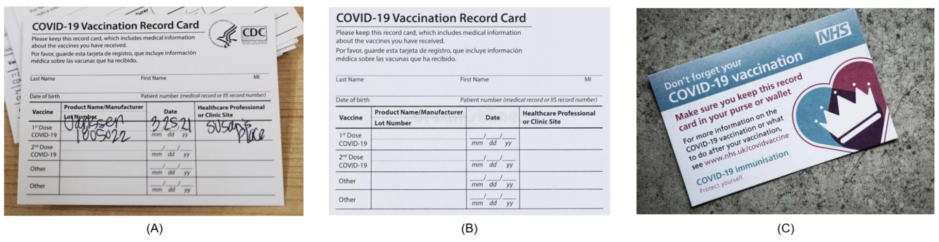
COVID-19 Vaccination Record Card
CDC
Please keep this record card, which includes medical information about the vaccines you have received.
Por favor, guarde esta tarjeta de registro, que incluye información médica sobre las vacunas que ha recibido.
Last Name
First Name
MI
Date of birth
Patient number (medical record or IIS record number)
Vaccine
Product Name/Manufacturer Lot Number
Date
Healthcare Professional or Clinic Site
1st Dose COVID-19
Janssen 1805022
3 25 21
Susan's Place
2nd Dose COVID-19
Other
mm dd yy
(A)
COVID-19 Vaccination Record Card
Please keep this record card, which includes medical information about the vaccines you have received.
Por favor, guarde esta tarjeta de registro, que incluye información médica sobre las vacunas que ha recibido.
Last Name
First Name
MI
Date of birth
Patient number (medical record or IIS record number)
Vaccine
Product Name/Manufacturer Lot Number
Date
Healthcare Professional or Clinic Site
1st Dose COVID-19
2nd Dose COVID-19
Other
mm dd yy
(B)
NHS
Don't forget your COVID-19 vaccination
Make sure you keep this record card in your purse or wallet
For more information on the COVID-19 vaccination or what to do after your vaccination, see www.nhs.uk/covidvaccine
COVID-19 immunisation
Protect yourself.
(C)

Figure 2: Different types of images. (A) a valid CDC-labeled COVID-19 vaccination card, (B) a vaccination card with missing feature (missing CDC logo), (C) a non-vaccination card.

All images were manually labelled by boxing 4 features/fields if they are present in the images, including i) vaccination card, ii) the header, iii) the CDC logo, and iv) the history of vaccination, resulting in a maximum of four boxes per image. We used an open-source annotation tool labelImg [14] for labelling, which outputs an "xml" file containing (xmin, ymin, xmax, ymax) for each box.

When working on the classification problem, we read the xml files and assigned the image whose label contains a boxed "card" object with "1", and the others with "0". Each image was resized, standardized, and represented by an (224, 224, 3) array.

When working on the detection problem, we transferred the annotations in XML files (where various attributes are described by tags) into YOLO v5 format .txt file (where each line of the text file describes a bounding box).

4 Approach

The idea is to build a simple binary classifier to first distinguish images showing vaccination card from those showing other types of card, and then only feed the vaccination card image into our feature detector downstream. Given a limited dataset, we expect higher efficiency (otherwise trying to capture the vaccination features on the irrelevant images would be time-consuming) and accuracy of this workflow compared with an end-to-end model.

To classify the images of vaccination cards, we built two classifiers, a simple CNN model with 5 hidden layers, and a CNN model built upon the pretrained VGG16 network. The first model was trained with all parameters trainable. The second model was trained by first replacing the final layer (a fully connected layer) of the pretrained VGG16 network with two additional layers (with "ReLu" and "Sigmoid" as activation functions) and then we froze all previous layers and only make the last two newly added layers trainable, resulting in 401,441 trainable parameters. The architectures of the two models are summarized in the Figure 3. For this classification task, 30% of data was held as the test set, and the remaining data was split into the training and validation set with a ratio of 7:3. To avoid a potential overfitting problem resulted from training for too many epochs, we borrowed the concept of early stopping, and plotted both the training error and validation error by the number of epochs to choose an optimal number of epoch. Adam optimization algorithm and cross entropy loss function were used in the training. Model performances were compared based on test accuracy.

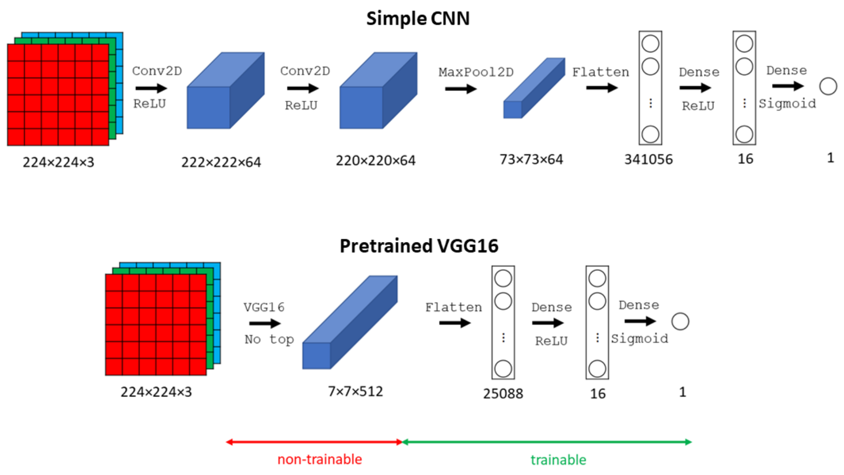

Figure 3: The two models used for card classification.

After picking out an image showing a vaccination card, we next detect the key features of the card. We chose the YOLOv5 algorithm for this task since it looks at the entire image and uses the global context to make predictions at test time. In addition, it is fast and has been proved to be a real time detector. The YOLOv5 family consists of a series of networks with different architectures. Of these, we tried three architectures, namely YOLOv5n (nano size, with 270 layers and 1.8 million parameters), YOLOv5s (small size, with 270 layers and 7.0 million parameters), and YOLOv5m (medium size, with 369 layers and 20.9 million parameters). The model was designed to detect 4 features/fields including card, header, logo, and history. A ratio of 8:1:1 was used to split the data (images showing a vaccination card) into training, validation and test set respectively. All images were resized into 640 by 640, and all models were trained for 100 epochs. Hyperparameter tuning was conducted for initial learning rate and beta values for the momentum term in the gradient calculation. Hyperparameters were chosen on a log scale (0.05, 0.01, 0.005, 0.001 for learning rate and 0.9, 0.99, 0.999 for beta). Model performances were compared primarily based on precision since we believe that it is more crucial to avoid false positive in a vaccination card detection task. When two models show close recall, we used recall as a complementary performance metric.

5 Results

5.1 Card Classification

Figure 4 shows training and validation error versus the number of epochs. As shown, although the training error kept decreasing as the number of epochs increased, the validation error first decreased and then increased with a turning point at around 10 epochs, indicating that overfitting occurred beyond 10 epochs' training. Therefore, we chose 8 as the number of epochs for model training.

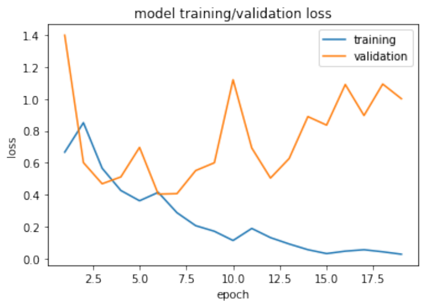

Figure 4: The evolution of training/validation error with number of epochs.

Table 1 shows the performance of two classifiers. We can see the CNN with pretrained VGG16 outperformed the simple CNN model by improving the test accuracy from 0.8879 to 0.9828. Therefore, the CNN with pretrained VGG16 model was chosen as our final card classifier.

Table 1. Performance of CNN Classifiers

	Training set		**Validation set**		**Test set**	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
CNN	0.1915	0.9403	0.4887	0.8382	0.3186	0.8879
CNN with VGG16	0.0036	1	0.2061	0.0.9265	0.0817	0.9828

5.2 Features Detection

Table 2 shows the performance of YOLOv5 with different architectures, initial learning rates and beta values ("P" stands for "precision", "R" stands for "recall", and "D" stands for "default value").

Table 2. Performance of YOLOv5 Models with Different Architectures/Hyperparameters

	CARD		**HEADR**		**LOGO**		**HISTORY**		**ALL**	
	P	R	P	R	P	R	P	R	P	R
					Different Architectures					
YOLOv5n	0.968	1	1	0.823	0.956	1	0.983	1	0.977	0.956
YOLOv5s	0.979	1	1	0.899	1	1	0.971	1	0.987	0.975
YOLOv5m	0.977	1	1	0.941	0.985	1	0.984	1	0.987	0.985
					Different Value for Initial Learning Rate					
0.001	0.752	1	0.999	0.706	0.762	1	0.698	0.941	0.803	0.912
0.005	0.985	1	1	0.886	0.974	1	1	1	0.99	0.971
0.01 (D)	0.977	1	1	0.941	0.985	1	0.984	1	0.987	0.985
0.05	0.979	1	1	1	0.979	1	0.969	1	0.982	1
					Different Value for Beta (in Momentum Term)					
0.9	0.965	1	1	1	0.961	1	0.958	1	0.971	1
0.937 (D)	0.977	1	1	0.941	0.985	1	0.984	1	0.987	0.985
0.99	0.977	1	1	0.941	0.974	1	0.975	1	0.982	0.985
0.999	0.975	1	1	0.941	0.981	1	0.981	1	0.984	0.985

First, we trained different architectures with fixed default learning rate (0.01) and beta (0.937), and found that YOLOv5m outperformed others in terms of overall precision (0.987) and recall (0.985). Next, we fixed YOLOv5m algorithm and trained with different initial learning rates. Although a learning rate of 0.005 showed the best overall precision, it performs poorly on correctly detecting header (R=0.886). By contrast, a learning rate of 0.01 showed a slightly lower overall precision but substantially improved the recall for header and overall. Therefore, we chose a learning rate of 0.01. Finally, we tuned the beta values and it turned out the default value of 0.937 achieved the best overall precision. Therefore, the optimal model would be a YOLOv5m algorithm with an initial learning rate of 0.01 and a beta value of 0.937. The change of loss, precision, recall, and F1 with the number of epochs during training have been plotted in the Figure 5 for this optimal model.

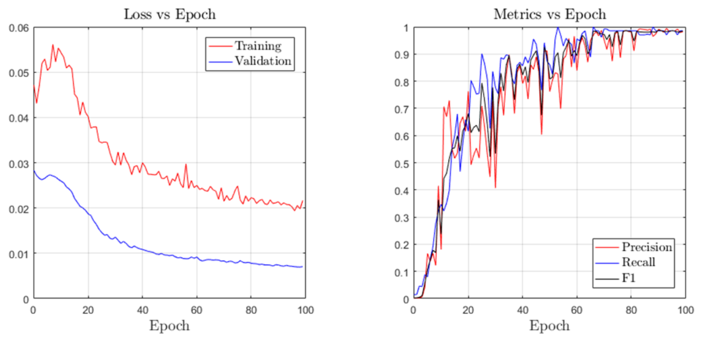

Figure 5: The evolution of different metrics with the number of epochs.

6 Discussion and Future Work

In this project, we developed an image classifier and an object detector which working together can achieve the classification and detection of the images of COVID-19 vaccination card.

There are several points worth noting. First, our classifier training verifies the advantage of using transfer learning (incorporating a model pretrained on a big dataset) in respect of improving the performance and dealing with a limited amount of data.

Second, a good learning rate must be discovered via trial and tuning on a log scale was efficient. In detection task, We also tried a learning rate of 0.1 while the algorithm was not able to converge, which may indicate a potential problem of exploding gradient (overshooting the minima).

Third, an error analysis found that a small model (YOLOv5s) failed to detect the "header" in 16% test images (upper plots in Figure 6). We think that the "header" might be a difficult feature to detect as it was defined by a sequence of words without any borders. In comparison, the "card", "logo", and "history" were bounded by solid rectangles or circles, making them easier to be recognized. A larger network (YOLOv5m) solved this issue and successfully recognized the "header" field in all test images, indicating a larger network may better exploit the inputs and capture complex features.

Additionally, even though a larger model was able to successfully detect all features, the predicted bounding boxes were not as accurate as the manually labeled ones. As shown by the two images at bottom of Figure 6, the predicted bounding box for the "card" is much smaller than the real box, leaving half of "header" outside the "card".

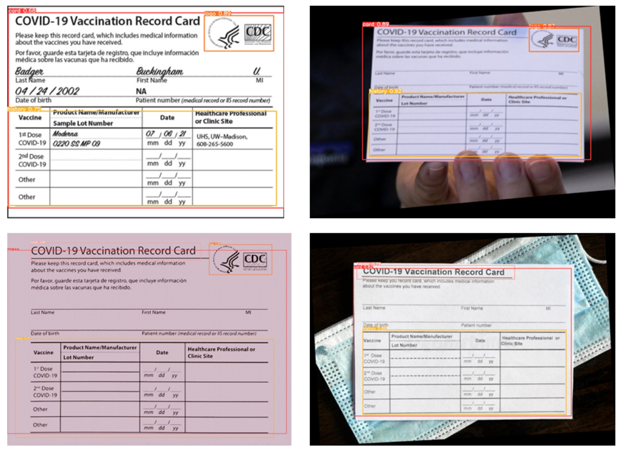
COVID-19 Vaccination Record Card
COVID-19 Vaccination Record Card
COVID-19 Vaccination Record Card
COVID-19 Vaccination Record Card

Figure 6: Predictions with missing "header" features (upper left and right) and predictions with inaccurate bounding box sizes (bottom left and right).

Future work may extend the current model to accommodate more features for online image verification or involve other algorithms to improve the model performance.

Appendix A Code

Please refer to Github directory `https://github.com/liuyx211/CS230-Project`.

Appendix B Member Contributions

Yingxiao worked on the implementation of the classifier and detector. Xiaojuan worked on image collection and hyperparameter tuning. We both worked on the data labelling/processing, results interpretation and report writing.

Appendix C More Results

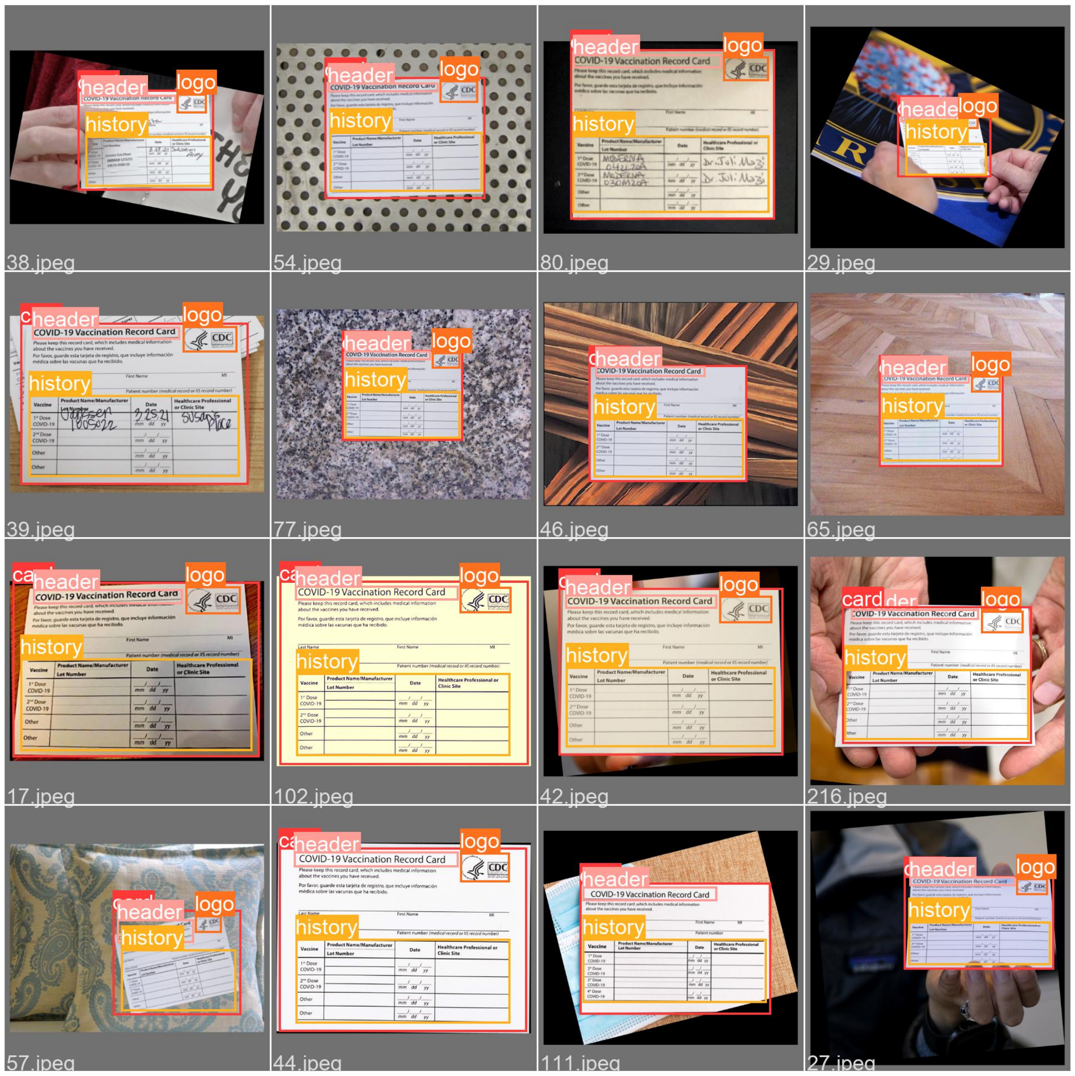

Figure 7: Selected images from the validation set showing manual labels.

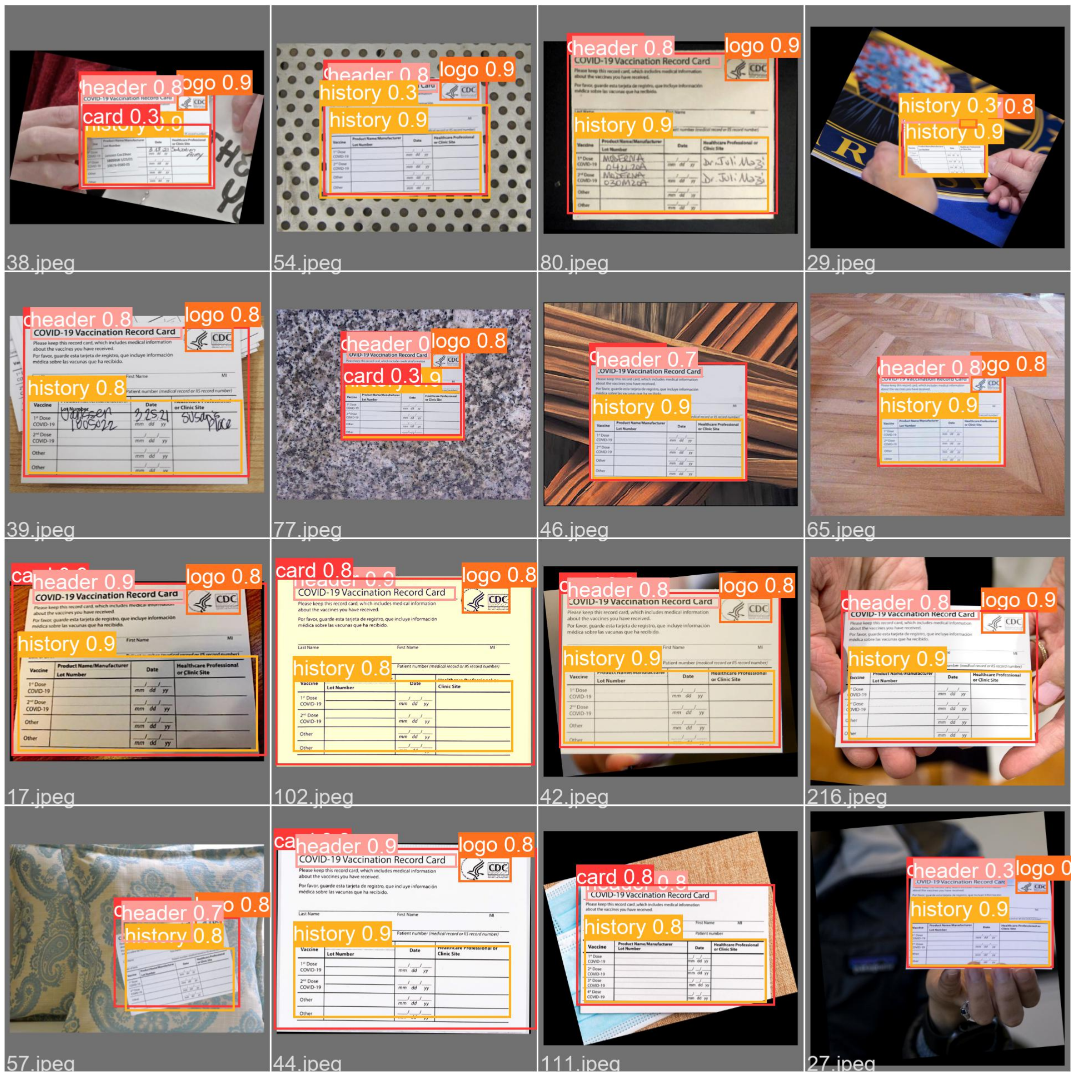

Figure 8: Same images from the validation set showing predicted labels by our model.

References

[1] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678).

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[3] Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7291-7299).

[4] Yang, Z.,& Nevatia, R. (2016, December). A multi-scale cascade fully convolutional network face detector. In 2016 23rd International Conference on Pattern Recognition (ICPR) (pp. 633-638). IEEE.

[5] Chen, C., Seff, A., Kornhauser, A., & Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE international conference on computer vision (pp. 2722-2730).

[6] Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1907-1915).

[7] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[8] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149.

[10] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).

[11] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[12] Tsai, C. M., Hsieh, J. W., Chang, M. C., & Lin, Y. C. (2020, September). Driver license field detection using real-time deep networks. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 603-613). Springer, Cham.

[13] Driver's License Data extraction using CNN (yolov5). `https://medium.com/analytics-vidhya/drivers-license-data-extraction-using-cnn-yolov5-14585709f4d8`

[14] LabelImg. `https://github.com/tzutalin/labelImg`