Overview:

From our research Tabula looks like a great candidate to benefit from improvements through machine learning. As pattern recognition in images can be achieved with great results using machine learning and Convnets. Future development could also investigate Natural Language processing as a means of further improving the output once cells are detected allowing for a greater understanding of the data and how it should be handled.

Recommendations for Machine Learning solution:

1) Use the framework called "Pytorch" for machine learning model deployment

- Pytorch is an open-source machine learning library for Python, based on the Torch library, used for applications such as computer vision

2) Use convolutional neural networks to create the machine learning model

- Convolutional neural networks (CNN) are artificial neural networks that are used for analysing images by detecting patterns in them. They can be used for other data analysis problems as well
- Basis of CNN are convolutional layers that receive the input, transforms the input and then outputs the transformed input
- This method can recognize straight edges (Figure 1) which is very useful for detecting tables in PDF documents

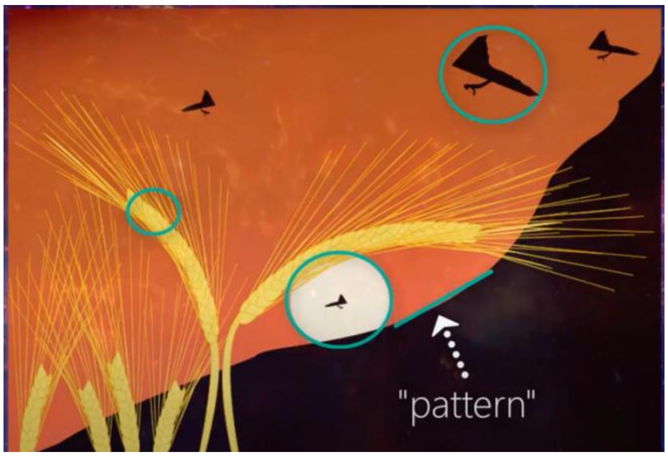


Figure 1: An edge is being detected as a pattern

3) Build from scratch

- We recommend building the Machine Learning project separate to the current iteration of the Tabula project as integrating the two would add extra work developing a communication method between the two systems and not provide as much benefit. Although there could be some advantage to detecting table positions with a CNN and passing that data to some form of the currently established Tabula system

4) Comparison of machine learning techniques and why CNN is best approach for this project

- Machine learning is split into two sections known as Unsupervised Learning and Supervised learning (Figure 2). Approach considered for this project would be supervised learning since it develops predictive model based on both input and output data.
- Supervised learning creates a model that gets trained to make predictions using a known set of input data and known responses to the data. The two categories are Classification and Regression (Figure 2). Both techniques have neural network as a common algorithm.
- Table below highlights the comparison:

Technique name	**What is it**	**Disadvantages compared to CNN**
Bootstrap Aggregation (Bagging)	Simple and powerful technique that combines the predictions from multiple machine learning algorithms for accurate predictions	Number of samples and number of trees to include. A large number of models can take long time to prepare.
Random Forests	A learning technique that creates multiple decision trees at training time and outputs the class that is the mean prediction of the individual trees	Training large set of deep trees can use a lot of memory and resources. Training time is longer since it generates plenty of trees to make decision.
K-Nearest Neighbors (KNN)	It is a simple algorithm that stores all available cases and organises new cases based on similarity measure. It assumes that similar things are near each other closely.	Calculating the distance between new point and existing points is very big, leading to a degrade in performance of the algorithm. It is sensitive to noisy data, missing values and outliners, highlighting how they need to be manually removed.
Support Vector Machine (SVM)	The algorithm finds a hyperplane (decision boundaries which help classify data points) in an N-dimensional space (N is the number of features) that categorises the data points.	Not efficient as it can't handle large data sets. Also takes a long time to train data sets. It doesn't work well when data set has more noise.

- In comparison to all of the listed machine learning techniques, CNN is the recommended option as it is computationally efficient, can adjust to unseen data very well and accommodate very large data sets unlike the other techniques listed above
- They can effectively reduce the number of parameters while maintaining high quality on the images.
- They are specially trained to identify edges of the objects in any image which is very important for this project as table identification (which has edges) is one of the first steps of the data extraction process.

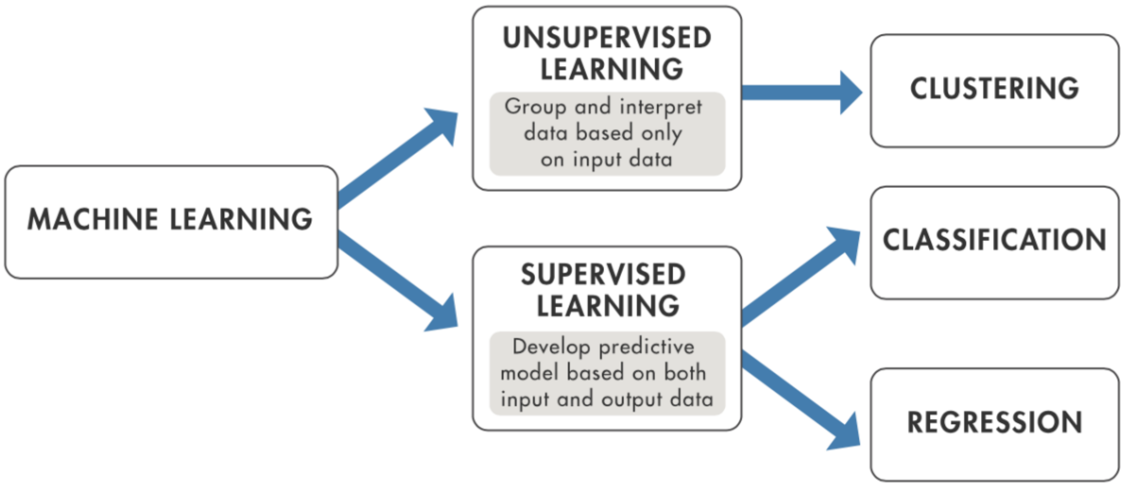

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Figure 2: Machine learning categories

5) Basics of how CNN works

- Pixels of an image is placed into CNN as an input in the form of arrays. Each block of the array Is labelled. The hidden layer reorganizes the image in multiple ways until the data that is easy to read for the neural network is found. This layer uses a matrix filter and there are multiple hidden layers in it such as the Convolutional layer and Pooling layer. Pooling layer uses multiple filters to detect edges and corners. Also, the image is transformed to be recognized in pixel form of 0's and 1's by the machine. The output layer is known as the fully connected layer which identifies the object in the image.

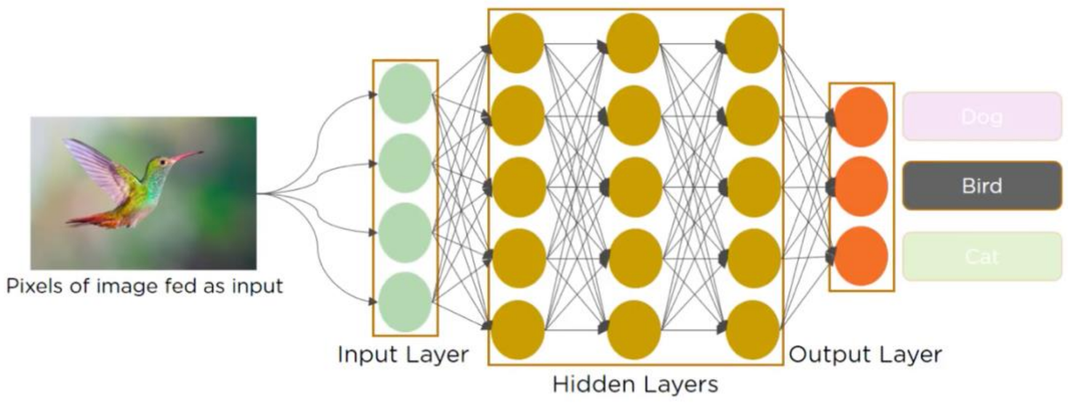


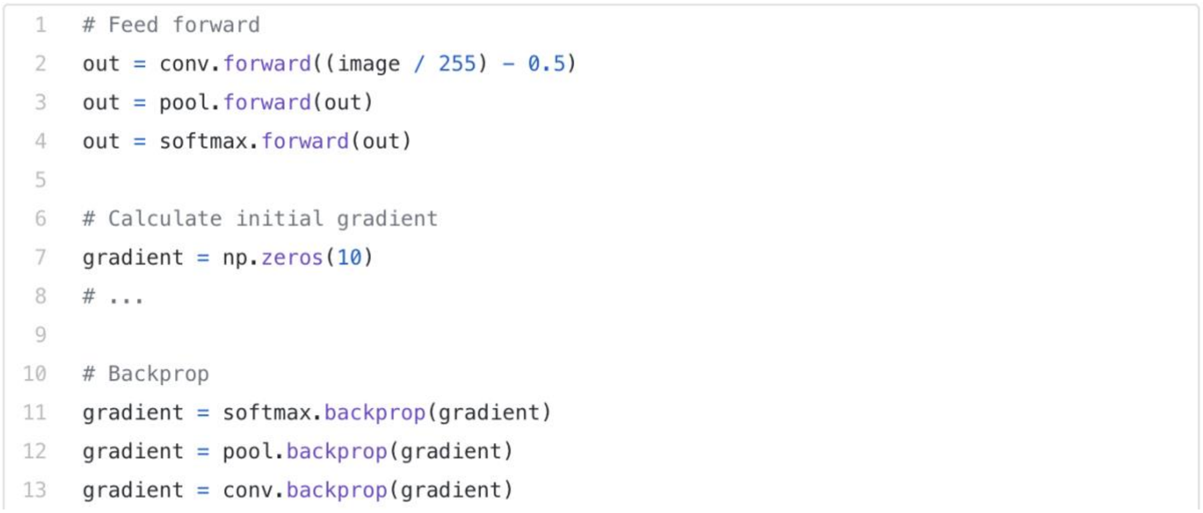
Figure 3: Basic process of CNN

6) Why CNN is very popular for image classification

- CNN utilizes some features of the visual cortex in the brain
- It can handle complex combination of patterns and repeatedly learn them until the image is recognized
- High level of accuracy
- Very efficient in terms of memory and efficiency
- It is fast and does not lose its quality
- Overfitting does not affect them greatly
- Have the "spatial invariance" property which enables them to identify image features anywhere on the chosen image. CNN is not sensitive to the position of the object in the image due to its ability to dive deeply into layers where similar objects become more similar

7) Information on training CNN

- There are two phases in training which are called the "forward" phase and the "backward" phase. In the "forward" phase, the input is passed thoroughly through the network and in the "backward" phase, the gradients are backpropagated and the weights are updated
- Additionally, there are two key implementation-specific ideas used which helps in maintaining the training stages organized. In the forward phase, each layer will cache data, such as inputs, it needs for the backward phase. In the backward phase, each layer will receive a gradient and return a gradient too.



```python
# Feed forward
out = conv.forward((image / 255) - 0.5)
out = pool.forward(out)
out = softmax.forward(out)

# Calculate initial gradient
gradient = np.zeros(10)
# ...

# Backprop
gradient = softmax.backprop(gradient)
gradient = pool.backprop(gradient)
gradient = conv.backprop(gradient)
```

Figure 4: Phases in coding form

8) Edge detection example

- There are plenty of vertical and horizontal edges in an image (Figure 5)

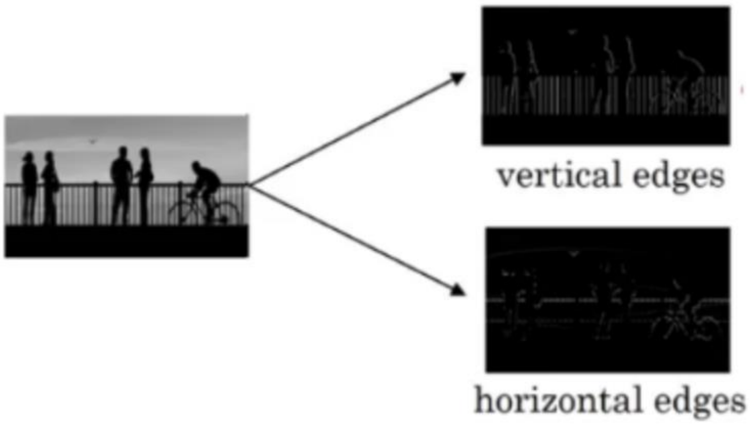
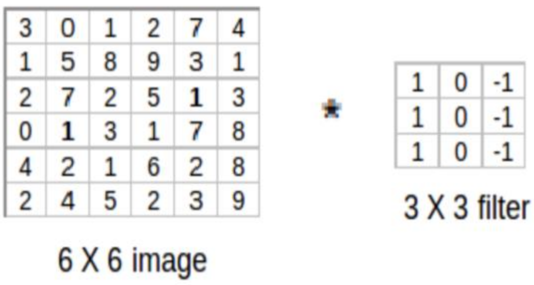


Figure 5: Types of edges in the particular image

- To detect these edges, a 6x6 matrix is taken and convolved with 3x3 filter (Figure 6). The result of the convolution is the 4x4 matrix, where the first element of the 4x4 matrix will be calculated as shown in Figure 7.



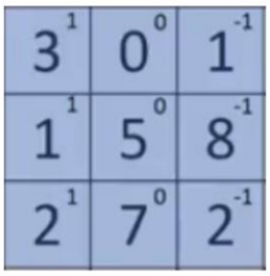
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 X 6 image

*

1	0	-1
1	0	-1
1	0	-1

3 X 3 filter

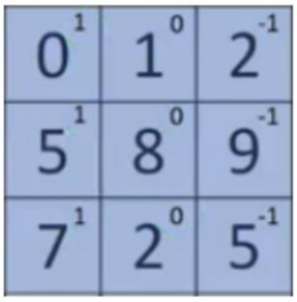


3^{1}	0^{0}	1^{-1}
1^{1}	5^{0}	8^{-1}
2^{1}	7^{0}	2^{-1}

Figure 6: Convolving matrix's

Figure 7: First element

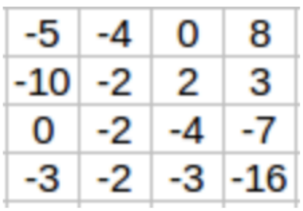
- First element of the 4x4 output is the sum of the element-wise product of these values: 3*1 + 0 + 1*-1 + 1*1 + 5*0 + 8*-1 + 2*1 + 7*0 + 2*-1 = -5
- To calculate the second element of the 4x4 output, shift the filter one step towards the right and get the sum of the element-wise product again



0^{1}	1^{0}	2^{-1}
5^{1}	8^{0}	9^{-1}
7^{1}	2^{0}	5^{-1}

Figure 8: Second element

- The whole image is then convolved and a 4x4 output is attained



-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Figure 9: Convolve entire image

- Here is the representation in image form. Higher pixel values are the brighter parts of the image and the lower pixel values are the darker parts. This is a way of identifying a vertical edge in an image.

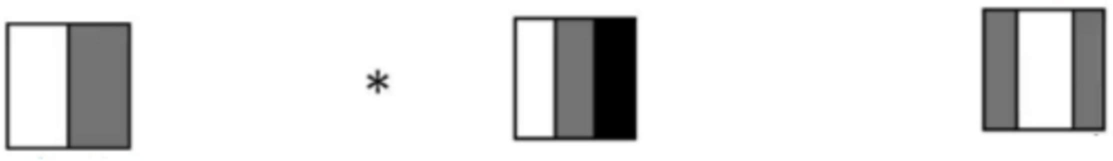


Figure 10: Vertical edge identification

Bibliography

Bonner, A., 2019. *The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification.* [Online]
Available at: https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb
[Accessed Wednesday June 2020].
Convolutional Neural Networks (CNNs) explained. 2017. [Film] Directed by deeplizard. s.l.: s.n.
Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm.* [Online]
Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
[Accessed Wednesday June 2020].
Introduction - Deep Learning and Neural Networks with Python and Pytorch p.1. 2019. [Film] Directed by sentdex. s.l.: s.n.
K, D., 2019. *Top 4 advantages and disadvantages of Support Vector Machine or SVM.* [Online]
Available at: https://medium.com/@dhiraj8899/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107
[Accessed Wednesday June 2020].
Mishra, P., 2019. *Why are Convolutional Neural Networks good for image classification?.* [Online]
Available at: https://medium.com/datadriveninvestor/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8
[Accessed Wednesday June 2020].
Sharma, P., 2018. *A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch (deeplearning.ai Course #4).* [Online]
Available at: https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
[Accessed Wednesday June 2020].
Shubham, J., 2018. *Ensemble Learning — Bagging and Boosting.* [Online]
Available at: https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e
[Accessed Wednesday June 2020].

Stecanella, B., 2017. *An Introduction to Support Vector Machines (SVM).* [Online]
Available at: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
[Accessed Wednesday June 2020].
Zhou, V., 2019. *Training a Convolutional Neural Network from scratch.* [Online]
Available at: https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754
[Accessed Wednesday June 2020].