# SVHN Digit Recognition

Deep Learning

## Contents / Agenda

## Business Problem Overview

- Description of the business problem that the analysis aims to solve
- Explanation of how the analysis can help achieve the business objective
- Mention the solution approach/methodology

### Description of the business problem that the analysis aims to solve

Currently, Google owns one of the most widely used geolocation applications worldwide, Google Maps. This makes the application not only interesting for users but also for the company, which can enhance its business and revenues around the app by incorporating new services. In this context, to enable users to quickly locate their searches through the application, it is crucial that the tool offers the possibility to accurately find the desired address. For this to happen, it is necessary for the application to be capable of recognizing and quickly locating the address number sought by the user.

With this in mind, the proposal emerged to develop a deep learning project for image recognition to assist the application in accurately locating addresses, as we will discuss further ahead.

### Explanation of how the analysis can help achieve the business objective

The deep learning approach for image recognition is widely recognized and adopted by companies that need to work with image recognition, regardless of their objectives. Whether for security requirements such as unlocking devices or allowing access to restricted areas, sentiment analysis, training autonomous vehicles to recognize and classify images on roads, or, as in our case, enabling the application to accurately recognize digits and thereby locate the desired user's address. This project will focus on developing a model that can recognize digits from 0 to 9, providing users with the assurance that their search will be successful. Achieving this goal will allow the company to satisfy its customers and add value to its application and business.

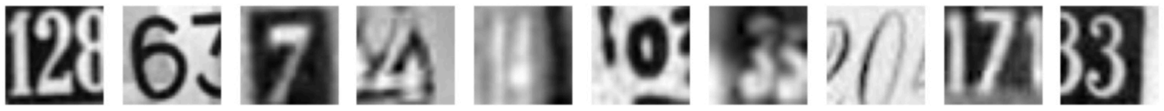
- **Mention the solution approach/methodology**

The chosen approach will involve the application of a deep learning model, utilizing a dataset of 60,000 digit images. This is essential for properly training the program so that it can easily and accurately recognize the numbers searched by the application users.

Our methodology will include the application of various image classification architectures for training and analyzing the selected database. This approach aims to determine the most efficient and effective architecture by the end of the project, which can then be implemented and utilized by the company.

## Data Overview

- Give an Overview of the Dataset



```
label for each of the above image: [2 6 7 4 4 0 3 0 7 3]
```

Above, we present a small sample of the dataset provided by the company, which will be employed in this project. The database consists of 60,000 images cropped from street-level photos, with 42,000 images utilized for model training and 18,000 for validation testing.

During the pre-processing stage, each image will be normalized. This involves transforming each label into numerical values, and subsequently, employing the one-hot encoding method to convert these categories into binary columns. These binary columns will be utilized in the model for image training. The idea behind one-hot encoding is to convert categorical variables (images) into variables that contain only 0s and 1s, as illustrated in the image below.

```
First image:
 [[ 33.0704   30.2601   26.852   ...   71.4471   58.2204   42.9939]
 [ 25.2283   25.5533   29.9765 ... 113.0209 103.3639   84.2949]
 [ 26.2775   22.6137   40.4763 ... 113.3028 121.775   115.4228]
 ...
 [ 28.5502   36.212    45.0801 ...   24.1359   25.0927   26.0603]
 [ 38.4352   26.4733   23.2717 ...   28.1094   29.4683   30.0661]
 [ 50.2984   26.0773   24.0389 ...   49.6682   50.853    53.0377]]
```

## Data Pre-Processing for ANNs

- Mention the data preprocessing steps required for making the training data usable for ANNs.
- Mention data preprocessing steps required for the labels

**Regarding the preprocessing steps, we can mention:**

**Label Encoding:**

In this step, we will convert the different categories (digits from 0 to 9 in images) into numerical values, as demonstrated earlier.

**One-Hot Encoding:**

Transforming the categories into binary columns of 0s and 1s.

**Considerations for Multiclasses:**

Since we are working with digits from 0 to 9, we will opt for a multiclasses system. This approach can adapt to multiple classes by assigning unique numbers or creating binary columns for each class.

## Model Performance Summary

Artificial Neural Networks: ANN

● Mention the main point of differences between the two models built.

● Show the training versus validation accuracy plot for the model that you choose.

Different structures with varying numbers of layers were applied to the two models used. Not only that, but in addition to adding more layers with a higher number of nodes in the second model, a Dropout layer of 0.2 was also introduced. This was done to ensure a reduction in the likelihood of overfitting, thereby promoting better generalization of the neural network during its training.

```
Epoch 1/30
263/263 [==============================] - 3s 6ms/step - loss: 2.2252 - accuracy: 0.1509 - val_loss:
1.9664 - val_accuracy: 0.2519
Epoch 2/30
263/263 [==============================] - 1s 5ms/step - loss: 1.8386 - accuracy: 0.3176 - val_loss:
1.7061 - val_accuracy: 0.4006
Epoch 3/30
263/263 [==============================] - 1s 5ms/step - loss: 1.5828 - accuracy: 0.4545 - val_loss:
1.4641 - val_accuracy: 0.5013
Epoch 4/30
263/263 [==============================] - 1s 6ms/step - loss: 1.4321 - accuracy: 0.5156 - val_loss:
1.3886 - val_accuracy: 0.5317
Epoch 5/30
263/263 [==============================] - 2s 7ms/step - loss: 1.3190 - accuracy: 0.5650 - val_loss:
1.2739 - val_accuracy: 0.5871
Epoch 6/30
263/263 [==============================] - 2s 7ms/step - loss: 1.2497 - accuracy: 0.5939 - val_loss:
1.1994 - val_accuracy: 0.6154
Epoch 7/30
263/263 [==============================] - 2s 7ms/step - loss: 1.1922 - accuracy: 0.6199 - val_loss:
1.1445 - val_accuracy: 0.6394
Epoch 8/30
263/263 [==============================] - 2s 7ms/step - loss: 1.1232 - accuracy: 0.6439 - val_loss:
1.1137 - val_accuracy: 0.6474
Epoch 9/30
263/263 [==============================] - 1s 5ms/step - loss: 1.0798 - accuracy: 0.6599 - val_loss:
1.1168 - val_accuracy: 0.6500
Epoch 10/30
263/263 [==============================] - 1s 5ms/step - loss: 1.0374 - accuracy: 0.6729 - val_loss:
1.0866 - val_accuracy: 0.6540
Epoch 11/30
263/263 [==============================] - 1s 5ms/step - loss: 0.9922 - accuracy: 0.6880 - val_loss:
1.0122 - val_accuracy: 0.6879
Epoch 12/30
263/263 [==============================] - 1s 5ms/step - loss: 0.9470 - accuracy: 0.7051 - val_loss:
0.9904 - val_accuracy: 0.6930
Epoch 13/30
263/263 [==============================] - 1s 5ms/step - loss: 0.9220 - accuracy: 0.7137 - val_loss:
0.9177 - val_accuracy: 0.7171
Epoch 14/30
263/263 [==============================] - 1s 5ms/step - loss: 0.8908 - accuracy: 0.7228 - val_loss:
0.9343 - val_accuracy: 0.7095
Epoch 15/30
263/263 [==============================] - 1s 5ms/step - loss: 0.8670 - accuracy: 0.7307 - val_loss:
0.8881 - val_accuracy: 0.7312
Epoch 16/30
263/263 [==============================] - 2s 7ms/step - loss: 0.8435 - accuracy: 0.7387 - val_loss:
0.9175 - val_accuracy: 0.7114
Epoch 17/30
263/263 [==============================] - 2s 8ms/step - loss: 0.8217 - accuracy: 0.7452 - val_loss:
0.8700 - val_accuracy: 0.7342
Epoch 18/30
263/263 [==============================] - 2s 7ms/step - loss: 0.8087 - accuracy: 0.7492 - val_loss:
0.8647 - val_accuracy: 0.7340
Epoch 19/30
263/263 [==============================] - 1s 6ms/step - loss: 0.7908 - accuracy: 0.7519 - val_loss:
0.8451 - val_accuracy: 0.7383
Epoch 20/30
```

```
263/263 [==============================] - 1s 5ms/step - loss: 0.7754 - accuracy: 0.7602 - val_loss:
0.8181 - val_accuracy: 0.7481
Epoch 21/30
263/263 [==============================] - 1s 5ms/step - loss: 0.7581 - accuracy: 0.7611 - val_loss:
0.8094 - val_accuracy: 0.7457
Epoch 22/30
263/263 [==============================] - 1s 5ms/step - loss: 0.7535 - accuracy: 0.7654 - val_loss:
0.7879 - val_accuracy: 0.7588
Epoch 23/30
263/263 [==============================] - 1s 5ms/step - loss: 0.7312 - accuracy: 0.7721 - val_loss:
0.7762 - val_accuracy: 0.7623
Epoch 24/30
263/263 [==============================] - 1s 5ms/step - loss: 0.7184 - accuracy: 0.7765 - val_loss:
0.7849 - val_accuracy: 0.7612
Epoch 25/30
263/263 [==============================] - 1s 5ms/step - loss: 0.7107 - accuracy: 0.7759 - val_loss:
0.7882 - val_accuracy: 0.7593
Epoch 26/30
263/263 [==============================] - 2s 6ms/step - loss: 0.7046 - accuracy: 0.7780 - val_loss:
0.7725 - val_accuracy: 0.7693
Epoch 27/30
263/263 [==============================] - 2s 7ms/step - loss: 0.6841 - accuracy: 0.7864 - val_loss:
0.7512 - val_accuracy: 0.7707
Epoch 28/30
263/263 [==============================] - 2s 7ms/step - loss: 0.6804 - accuracy: 0.7867 - val_loss:
0.7725 - val_accuracy: 0.7642
Epoch 29/30
263/263 [==============================] - 2s 7ms/step - loss: 0.6715 - accuracy: 0.7896 - val_loss:
0.7623 - val_accuracy: 0.7723
Epoch 30/30
263/263 [==============================] - 1s 5ms/step - loss: 0.6510 - accuracy: 0.7962 - val_loss:
0.7574 - val_accuracy: 0.7720
```

We can observe from these results that with each generated epoch, the loss was reduced, and both accuracy and validation accuracy improved. Despite the good results obtained, we recognize that further improvements can be implemented. Consequently, we have decided to apply two convolutional neural network models, the results of which will be discussed below.

# Model Performance Summary

**Convolutional Neural Networks:**

- Mention the main point of differences between the two models built.
- Show the training versus validation accuracy plot for the model that you choose.
- Show precision and F1 score for each digit and also plot the Confusion matrix and explain a few findings.
- Write overall Observations

## Mention the main point of differences between the two models built.

Among the main differences between the two CNN architectures applied in this project, the following points can be highlighted:

**Additional Layers:**
More layers were added so that the model can train with more image details.

**Batch Normalization Layer:**
A BatchNormalization layer was introduced, which, when combined with other techniques, enhances the overall performance of the architecture.

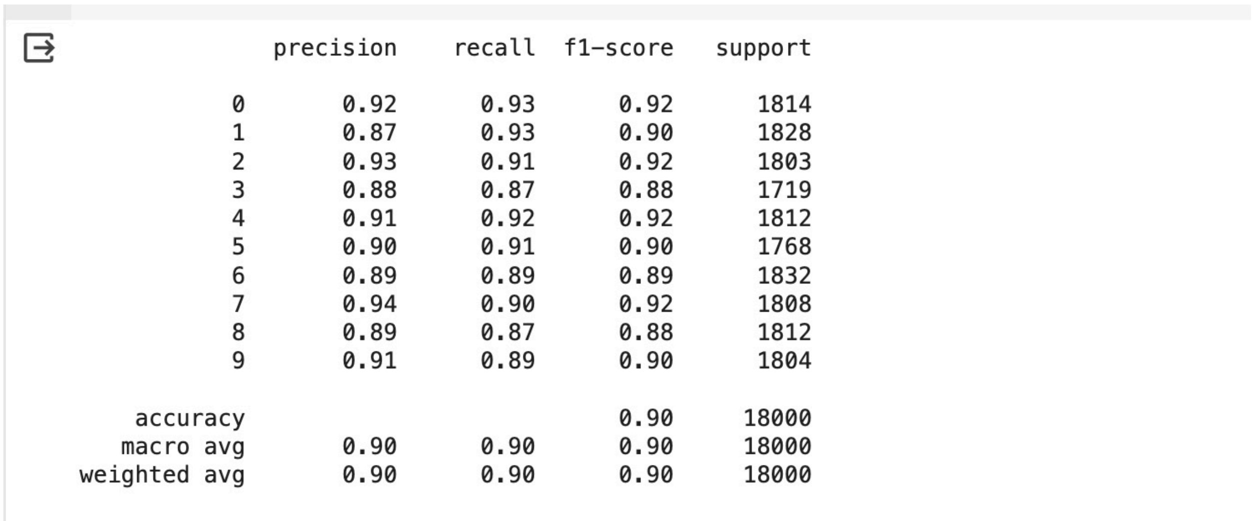
**Dropout Layer:**
A Dropout layer of 0.5 (50%) was added to address potential overfitting concerns.

**Show the training versus validation accuracy plot for the model that you choose.**

```
Epoch 1/30
263/263 [==============================] - 7s 15ms/step - loss: 2.1389 -
accuracy: 0.1845 - val_loss: 1.2290 - val_accuracy: 0.6773
Epoch 2/30
263/263 [==============================] - 3s 13ms/step - loss: 1.1805 -
accuracy: 0.5847 - val_loss: 0.5891 - val_accuracy: 0.8281
Epoch 3/30
263/263 [==============================] - 4s 15ms/step - loss: 0.9284 -
accuracy: 0.6770 - val_loss: 0.5141 - val_accuracy: 0.8526
Epoch 4/30
263/263 [==============================] - 3s 12ms/step - loss: 0.8339 -
accuracy: 0.7079 - val_loss: 0.4848 - val_accuracy: 0.8604
Epoch 5/30
263/263 [==============================] - 3s 11ms/step - loss: 0.7719 -
accuracy: 0.7300 - val_loss: 0.4418 - val_accuracy: 0.8725
Epoch 6/30
263/263 [==============================] - 3s 12ms/step - loss: 0.7361 -
accuracy: 0.7396 - val_loss: 0.4267 - val_accuracy: 0.8780
Epoch 7/30
263/263 [==============================] - 4s 13ms/step - loss: 0.7043 -
accuracy: 0.7482 - val_loss: 0.4102 - val_accuracy: 0.8787
Epoch 8/30
263/263 [==============================] - 4s 14ms/step - loss: 0.6761 -
accuracy: 0.7570 - val_loss: 0.4045 - val_accuracy: 0.8838
Epoch 9/30
263/263 [==============================] - 3s 13ms/step - loss: 0.6669 -
accuracy: 0.7602 - val_loss: 0.3960 - val_accuracy: 0.8855
Epoch 10/30
263/263 [==============================] - 3s 11ms/step - loss: 0.6464 -
accuracy: 0.7658 - val_loss: 0.3837 - val_accuracy: 0.8906
Epoch 11/30
263/263 [==============================] - 3s 11ms/step - loss: 0.6305 -
accuracy: 0.7718 - val_loss: 0.3967 - val_accuracy: 0.8851
Epoch 12/30
263/263 [==============================] - 4s 14ms/step - loss: 0.6157 -
accuracy: 0.7781 - val_loss: 0.3804 - val_accuracy: 0.8888
Epoch 13/30
263/263 [==============================] - 4s 15ms/step - loss: 0.6045 -
accuracy: 0.7801 - val_loss: 0.3865 - val_accuracy: 0.8879
Epoch 14/30
263/263 [==============================] - 3s 13ms/step - loss: 0.5979 -
accuracy: 0.7813 - val_loss: 0.3872 - val_accuracy: 0.8874
Epoch 15/30
263/263 [==============================] - 3s 13ms/step - loss: 0.5611 -
accuracy: 0.7990 - val_loss: 0.3853 - val_accuracy: 0.8920
Epoch 16/30
263/263 [==============================] - 3s 11ms/step - loss: 0.5472 -
accuracy: 0.8067 - val_loss: 0.3665 - val_accuracy: 0.8985
```

```
Epoch 17/30
263/263 [==============================] - 4s 15ms/step - loss: 0.5365 -
accuracy: 0.8074 - val_loss: 0.3634 - val_accuracy: 0.8957
Epoch 18/30
263/263 [==============================] - 3s 13ms/step - loss: 0.5283 -
accuracy: 0.8090 - val_loss: 0.3632 - val_accuracy: 0.8980
Epoch 19/30
263/263 [==============================] - 3s 11ms/step - loss: 0.5172 -
accuracy: 0.8122 - val_loss: 0.3785 - val_accuracy: 0.8949
Epoch 20/30
263/263 [==============================] - 3s 11ms/step - loss: 0.5048 -
accuracy: 0.8172 - val_loss: 0.3738 - val_accuracy: 0.8988
Epoch 21/30
263/263 [==============================] - 3s 12ms/step - loss: 0.4900 -
accuracy: 0.8235 - val_loss: 0.3720 - val_accuracy: 0.9002
Epoch 22/30
263/263 [==============================] - 4s 14ms/step - loss: 0.4841 -
accuracy: 0.8220 - val_loss: 0.3713 - val_accuracy: 0.8968
Epoch 23/30
263/263 [==============================] - 4s 14ms/step - loss: 0.4874 -
accuracy: 0.8222 - val_loss: 0.3745 - val_accuracy: 0.8995
Epoch 24/30
263/263 [==============================] - 3s 11ms/step - loss: 0.4727 -
accuracy: 0.8265 - val_loss: 0.3649 - val_accuracy: 0.9024
Epoch 25/30
263/263 [==============================] - 3s 11ms/step - loss: 0.4673 -
accuracy: 0.8287 - val_loss: 0.3816 - val_accuracy: 0.8960
Epoch 26/30
263/263 [==============================] - 3s 11ms/step - loss: 0.4664 -
accuracy: 0.8291 - val_loss: 0.3724 - val_accuracy: 0.8998
Epoch 27/30
263/263 [==============================] - 4s 14ms/step - loss: 0.4609 -
accuracy: 0.8312 - val_loss: 0.3824 - val_accuracy: 0.9004
Epoch 28/30
263/263 [==============================] - 3s 13ms/step - loss: 0.4517 -
accuracy: 0.8296 - val_loss: 0.3706 - val_accuracy: 0.9000
Epoch 29/30
263/263 [==============================] - 3s 12ms/step - loss: 0.4387 -
accuracy: 0.8373 - val_loss: 0.3755 - val_accuracy: 0.9033
Epoch 30/30
263/263 [==============================] - 3s 13ms/step - loss: 0.4386 -
accuracy: 0.8350 - val_loss: 0.3858 - val_accuracy: 0.9025
```

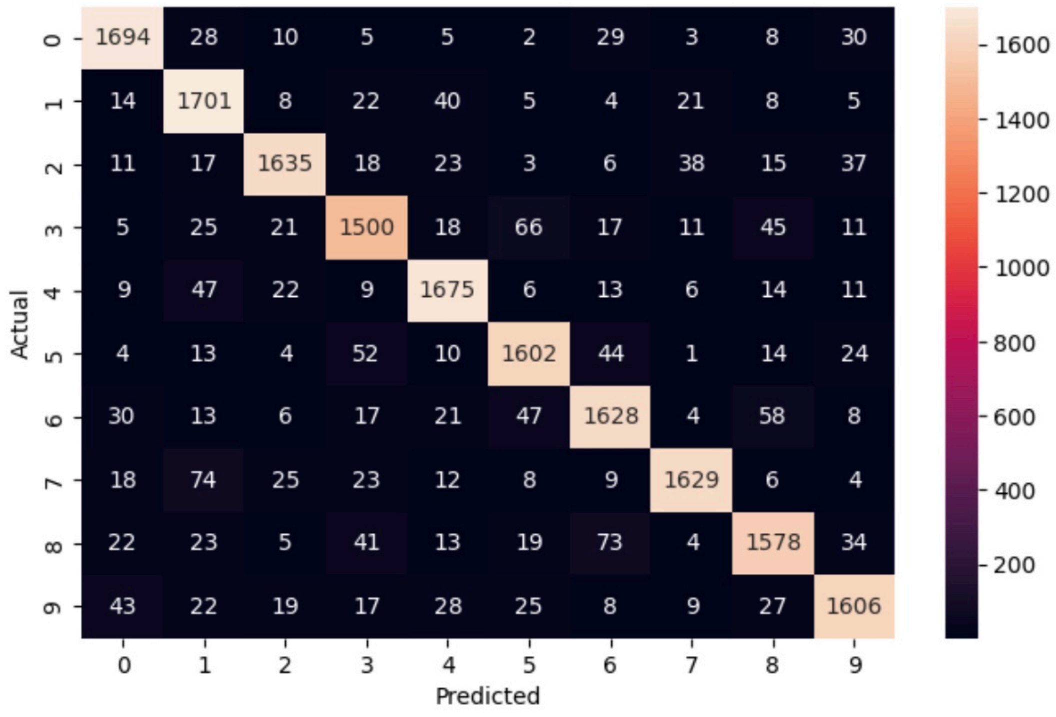
**Show precision and F1 score for each digit and also plot the Confusion matrix and explain a few findings.**



```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92      1814
           1       0.87      0.93      0.90      1828
           2       0.93      0.91      0.92      1803
           3       0.88      0.87      0.88      1719
           4       0.91      0.92      0.92      1812
           5       0.90      0.91      0.90      1768
           6       0.89      0.89      0.89      1832
           7       0.94      0.90      0.92      1808
           8       0.89      0.87      0.88      1812
           9       0.91      0.89      0.90      1804

    accuracy                           0.90     18000
   macro avg       0.90      0.90      0.90     18000
weighted avg       0.90      0.90      0.90     18000
```



| Actual \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1694 | 28 | 10 | 5 | 5 | 2 | 29 | 3 | 8 | 30 |
| 1 | 14 | 1701 | 8 | 22 | 40 | 5 | 4 | 21 | 8 | 5 |
| 2 | 11 | 17 | 1635 | 18 | 23 | 3 | 6 | 38 | 15 | 37 |
| 3 | 5 | 25 | 21 | 1500 | 18 | 66 | 17 | 11 | 45 | 11 |
| 4 | 9 | 47 | 22 | 9 | 1675 | 6 | 13 | 6 | 14 | 11 |
| 5 | 4 | 13 | 4 | 52 | 10 | 1602 | 44 | 1 | 14 | 24 |
| 6 | 30 | 13 | 6 | 17 | 21 | 47 | 1628 | 4 | 58 | 8 |
| 7 | 18 | 74 | 25 | 23 | 12 | 8 | 9 | 1629 | 6 | 4 |
| 8 | 22 | 23 | 5 | 41 | 13 | 19 | 73 | 4 | 1578 | 34 |
| 9 | 43 | 22 | 19 | 17 | 28 | 25 | 8 | 9 | 27 | 1606 |

As shown in the images above, we can observe that the accuracy of the proposed model reaches approximately 90%, which we consider a good result for the initially stated objective.

Upon closer inspection of our Classification Report, we notice that digit 7 achieved a precision of 94%, which is excellent.

This pattern is clearly highlighted in our Heat Map, where the number of correct predictions compared to their respective samples is notably high.

## Choosing the Final Model:

- Write about which approach you are choosing as the final solution and why.

In this case, we chose to adopt the second model as we consider it more comprehensive and yielding better final results for the proposed task. We believe that with the normalization of layers and the addition of the Dropout layer, the results were more consistent, leading to the presented outcomes, which, in our opinion, are quite satisfactory.

If the company wishes to explore further enhancements to the model, we can consider other approaches involving additional techniques to assess their impact on results. This may involve experimenting with different architectures to determine if they yield better results or if improvements become negligible compared to what has been presented so far. Additionally, expanding the dataset could be considered to explore the potential outcomes of applying the same model to a larger dataset.

## Appendix:

- Additional details on the analysis, such as code snippets or technical diagrams
- Any supplementary information or supporting materials that were used in the analysis