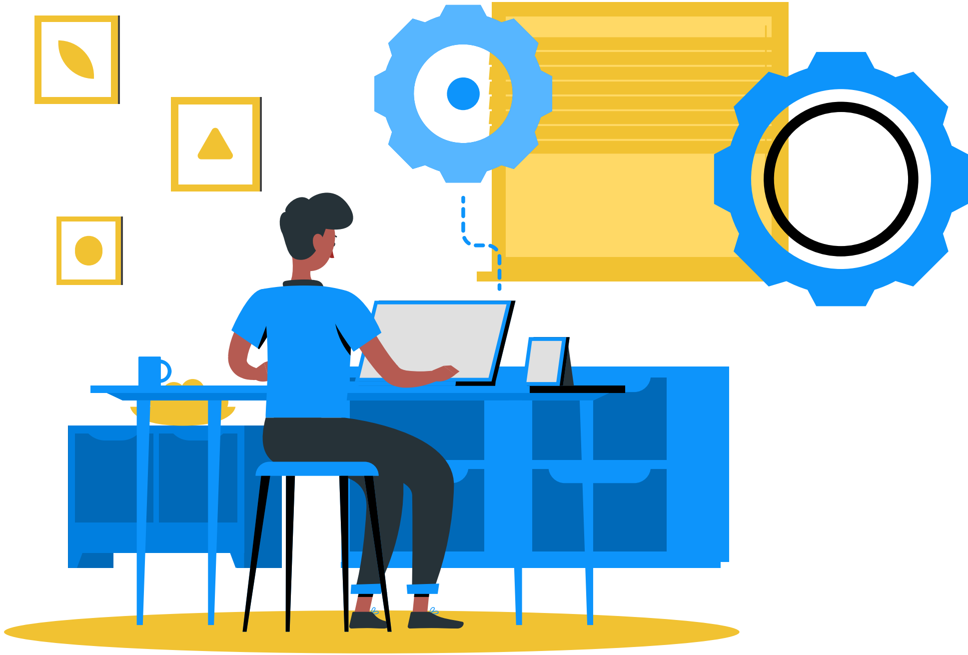
Universal Currency Recognition for the Visually Impaired

Khyati Jariwala, Nicolas Rey,
Raghav Vaidya, Rajkumar Rajavel, Yu Xia

Table of contents

Problem: Background

- Large population with **visual impairments**: low visibility, partial blindness, blindness and more
- World is **not** vision impairment friendly
- Difficulty in the domain of **recognizing different currency banknotes**
    - Hinders independence because money is involved in many daily transactions



Problem: Statistics

2.2B

People with a near or distance vision impairment

$411B

Estimated cost of productivity due to vision impairment

4x

Vision impairment in lower income regions vs high-income regions

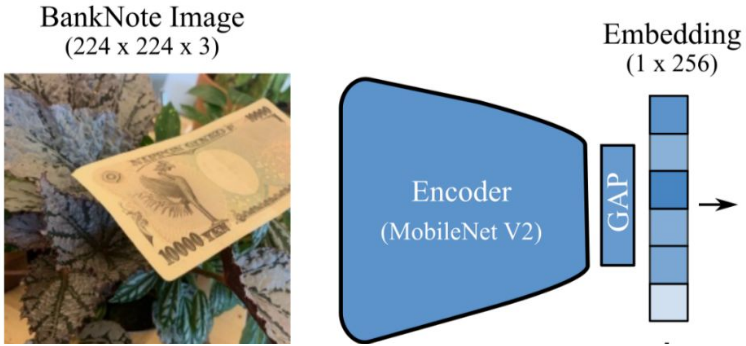
Problem: Goal

- **Objective:** To leverage predictive modeling to enable **universal currency recognition** to assist individuals with visual impairments

- **Our Approach:**
  - Use a **diverse dataset** encompassing many currency denominations
  - Train a **Neural Network** on the data
  - Assess the **performance** of the NN on currency

Data

- **Data Source:** Microsoft’s BankNote-Net embeddings (largest open dataset of banknote images for accessibility)
- **How were the embeddings obtained by Microsoft?**
  - **100+ images** of front and back of each denomination of each currency was taken in various real assistive scenarios (lighting conditions, orientation, etc.)
    - Considers **diversity and applicability** of models trained from dataset
  - Standardized images to **224 x 224 pixels**
  - MobileNet V2 was used as the **encoder** to obtain the embeddings
  - Global average pooling layer behaves similar to a max pooling layer but takes average of the features



Data

- **Data Structure:**
  - **Rows:** Each row corresponds to one image. The values across the columns form the embedding of the image
  - **Columns (v_0 to v_255):** Each column represents a feature extracted from the image
  - **'Currency'** and **'Denomination'** indicate the type and value of the banknote in the image, _1 and _2 denote the front and back of the notes
- By comparing these embeddings, we can quantify **similarities or differences** between images, aiding in predictive modeling and classification

v_254	v_255	Currency	Denomination	currency_denom
4.724614	0.000000	AUD	100_1	AUD_100_1
2.648906	0.656381	AUD	100_1	AUD_100_1
0.823947	1.539916	AUD	100_1	AUD_100_1
1.724243	0.000000	AUD	100_1	AUD_100_1
2.969594	0.000000	AUD	100_1	AUD_100_1
3.243896	0.000000	AUD	100_1	AUD_100_1
3.043229	0.000000	AUD	100_1	AUD_100_1
3.162983	0.043580	AUD	100_1	AUD_100_1
3.257974	0.000000	AUD	100_1	AUD_100_1
1.862063	0.000000	AUD	100_1	AUD_100_1
1.850240	1.104519	AUD	100_1	AUD_100_1
0.736805	0.955484	AUD	100_1	AUD_100_1
2.704934	0.000000	AUD	100_1	AUD_100_1
1.999638	0.000000	AUD	100_1	AUD_100_1
0.567682	1.242405	AUD	100_1	AUD_100_1

Data Description

24,826

Embeddings of banknote images

17

Types of currencies

112

Currency denominations

224

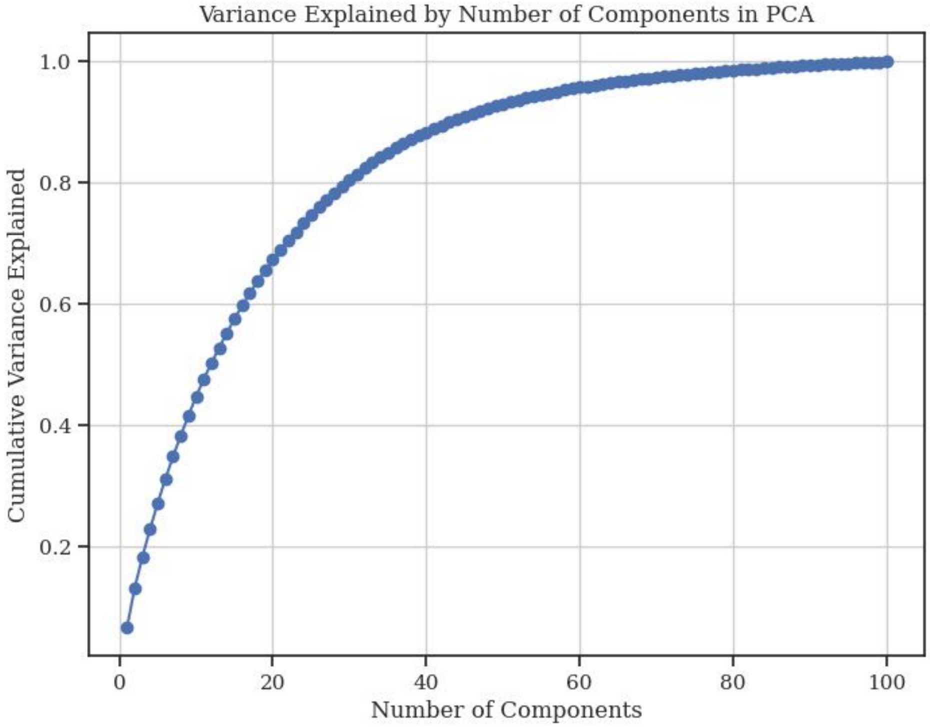
Classes

- **Most Represented Currencies:** Turkish Lira (TRY) and Brazilian Real (BRL)

```
data_df.Currency.value_counts()
```

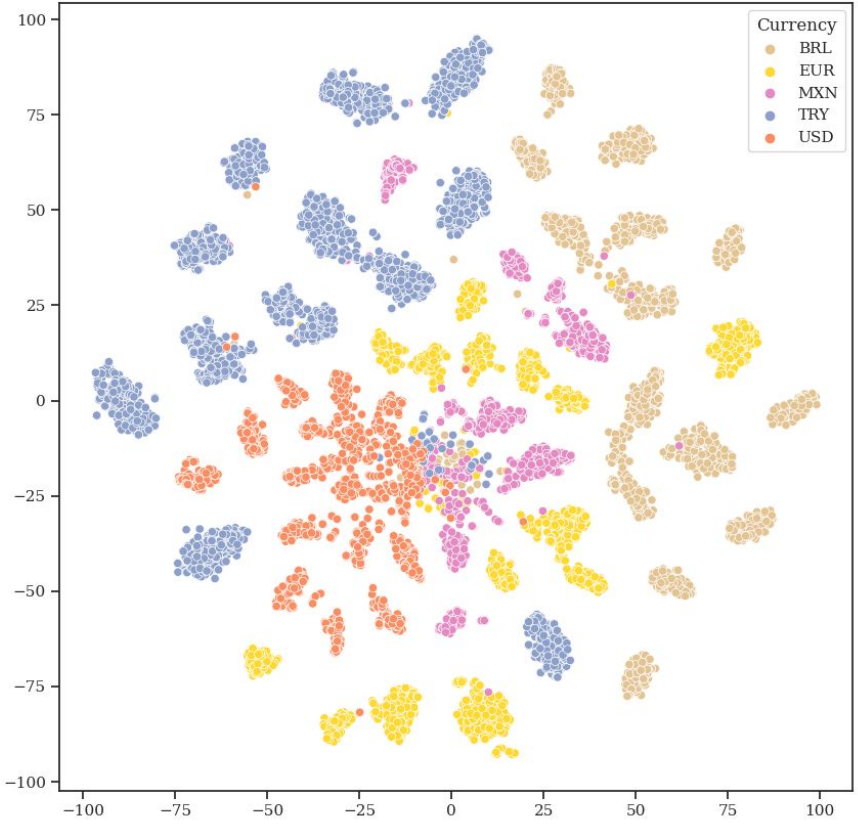
TRY	2888
BRL	2078
INR	1921
EUR	1905
JPY	1658
AUD	1616
USD	1604
MYR	1202
IDR	1164
PHP	1164
CAD	1162
NZD	1156
PKR	1131
MXN	1122
GBP	1108
SGD	1015
NNR	932

EDA: PCA



- **PCA performed with 100 components**
  - 40 components explain about **90% of the variance**

EDA: t-SNE

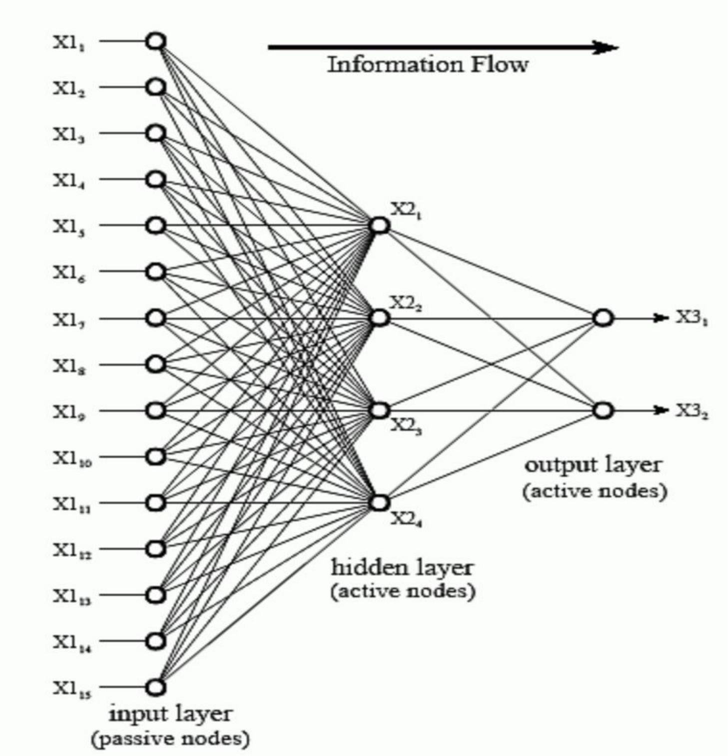


- **t-SNE showing embeddings of 5 currencies:** Brazilian Real, Euro, Mexican Peso, Turkish Lira, US Dollar
- Observe natural clustering of individual currencies
  - Color is currency and number of currencies are the number of classes within that currency
- Takes **fewer components** to visualize difference in classes compared to PCA

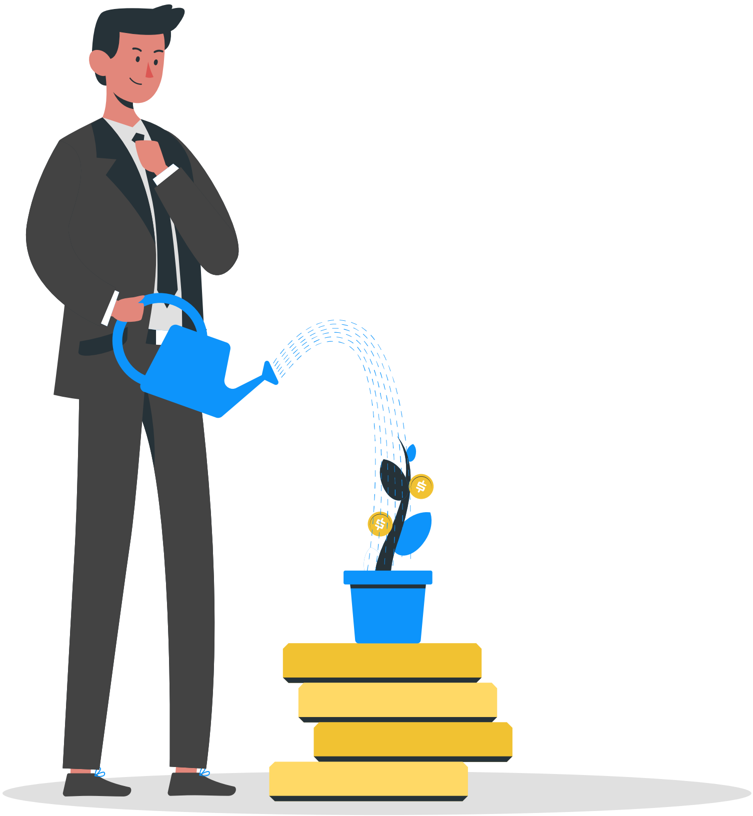
Modeling: Neural Network (NN)

How Neural Networks Work:

- **Input Layer:** Receives image embeddings
- **Hidden Layers:** Multiple layers where the actual processing is done via neurons. Each layer extracts increasingly complex features from the input
- **Output Layer:** Assigns probabilities of the embeddings for belonging to each of the x classes



Modeling: Neural Network (NN)



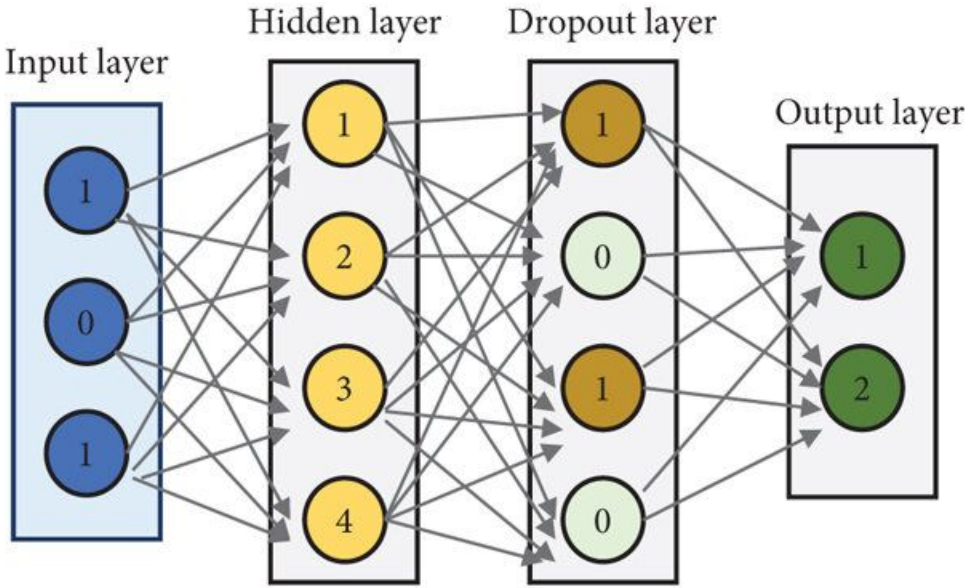
Why Neural Networks?:

- Good Performance predicting classes
- Efficiency in Feature Extraction

Why not CNN?

- Since we are working with embeddings as inputs and not actual images

Modeling: Creating the NN



Input layer: vectors of size 256

Dense layer: ReLU activation function with 128 neurons

Dropout layer: prevents overfitting

Dense layer: Softmax function to enable multi-class classification

Model Iteration	Validation Accuracy
1 layer - No dropout	94%
1 layer - 10% dropout	95%
2 layers - No dropout	93%
2 layers - 10% dropout	94%

- Batch size: 128
- Epochs: 25

```
Model: "model_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_22 (InputLayer)       [(None, 256)]             0

 dense_22 (Dense)            (None, 128)               32896

 dropout_11 (Dropout)        (None, 128)               0

 dense_23 (Dense)            (None, 14)                1806

=================================================================
Total params: 34,702
Trainable params: 34,702
Non-trainable params: 0
```

Results: Out of sample Performance Metrics

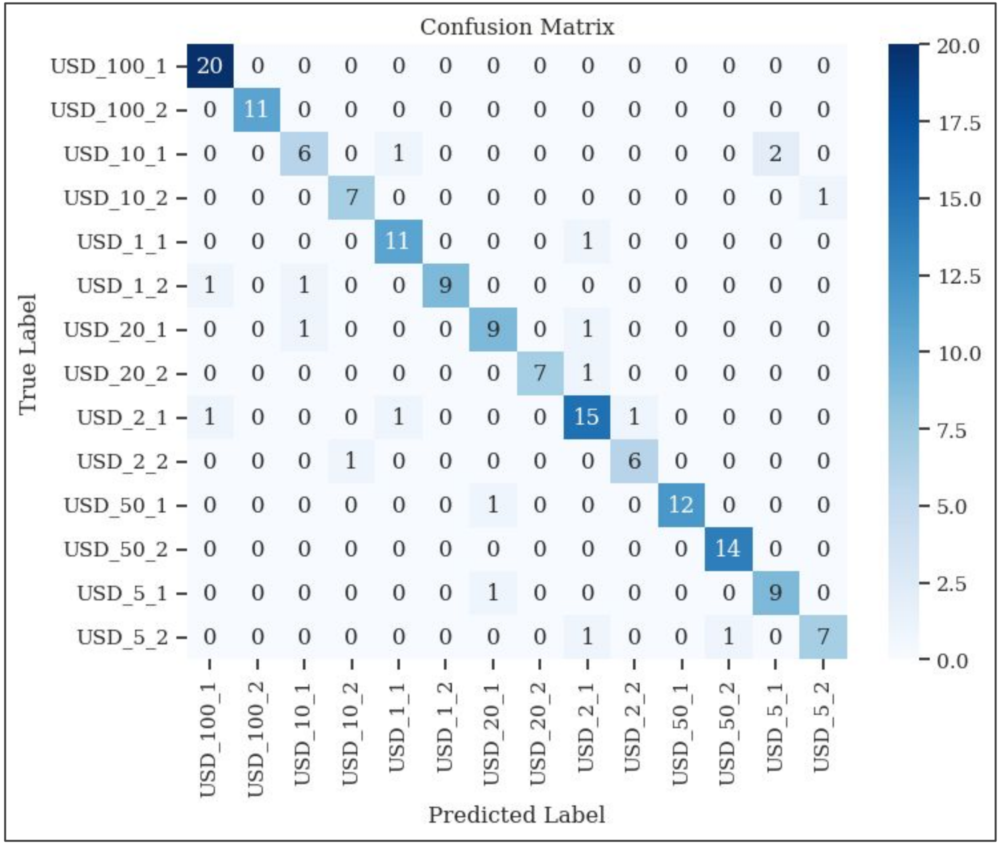
Accuracy: 0.8881987577639752

Classification Report:

	precision	recall	f1-score	support
0	0.91	1.00	0.95	20
1	1.00	1.00	1.00	11
2	0.75	0.67	0.71	9
3	0.88	0.88	0.88	8
4	0.85	0.92	0.88	12
5	1.00	0.82	0.90	11
6	0.82	0.82	0.82	11
7	1.00	0.88	0.93	8
8	0.79	0.83	0.81	18
9	0.86	0.86	0.86	7
10	1.00	0.92	0.96	13
11	0.93	1.00	0.97	14
12	0.82	0.90	0.86	10
13	0.88	0.78	0.82	9
accuracy			0.89	161
macro avg	0.89	0.88	0.88	161
weighted avg	0.89	0.89	0.89	161

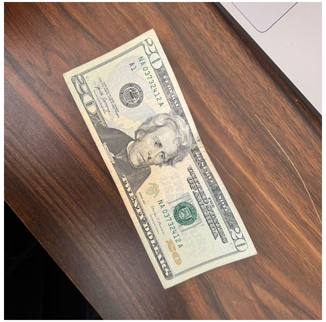
- **How well does our NN perform?**
  - **Accuracy:** 88.82%
  - **Precision:** (1, 5, 7, 10, 11) achieving perfect precision (1.00)
  - **Recall:** (0, 1, 11) having perfect recall (1.00)

Results: Out of sample Confusion Matrix



- Model shows **good performance** across different classes
  - The predicted label generally **matched** the true label of the USD

Results: Using Our Own Images



- **Custom Images:** We wanted to test the NN performance on images that we trained on USD currency
    - Interested in seeing if the NN could **predict the currency and denomination** given these images

Results: Using Our Own Images

```python
IMG_SIZE = (224, 224)

# Load the pre-trained encoder model
encoder_model = load_model("../models/banknote_net_encoder.h5")

# Input tensor with shape (IMG_SIZE[0], IMG_SIZE[1], 3)
input1 = Input(shape=(IMG_SIZE[0], IMG_SIZE[1], 3))

# Apply the encoder model to transform input1 to the desired shape (256,)
output_of_encoder = encoder_model(input1)

# Create a new model with the transformed input
model_with_transformed_input = Model(inputs=input1, outputs=output_of_enco


# Use the model to predict the class of the input
input_image = tf.keras.preprocessing.image.load_img("/Users/nicolasrey/Doc
input_image_array = tf.keras.preprocessing.image.img_to_array(input_image)
input_image_array = tf.expand_dims(input_image_array, axis=0)

# Transform the input using the model_with_transformed_input
transformed_input = model_with_transformed_input.predict(input_image_array

# Use the pre-trained "model" to predict the class
predictions = model.predict(transformed_input)
predictions = np.argmax(predictions, axis=1)
print("Predictions:")
print(predictions)
```

- **Steps:**
  1. Encode the custom images into **embeddings**
  2. Create a **new model** with the transformed input
  3. Use the NN model to **predict the class** of the currency
  4. Examine the **results**

Results: Using Our Own Images

Currency / Denomination / Class	Probability
USD_100_1	0.1662
USD_100_2	0.0080
USD_10_1	0.0395
USD_10_2	0.1979
USD_1_1	0.0608
USD_1_2	0.0037
USD_20_1	0.0299
USD_20_2	0.0017
USD_2_1	0.4140
USD_2_2	0.0638
USD_50_1	0.0016
USD_50_2	0.0009
USD_5_1	0.0060
USD_5_2	0.0059

Using our model, we were able to predict that the picture contains the **front of a 2 dollar bill with 41% probability** compared to other possible USD denomination

Limitations

1

Model Complexity: The current shallow model may not capture all complex patterns in the data

2

Data Imbalance: Some denominations more represented than others, potentially leading to bias

3

Lighting and Quality Variations: The model's performance can vary under different lighting condition or image qualities

Future Scope

Deep Learning Enhancements

Potentially improved feature extraction and accuracy.

Multi-angle Image Processing

Improve the robustness of the model against different orientations.

Increase the Dataset size

Can help the model generalize better to real-world variations.

Real-time Processing

For use in mobile applications for the visually impaired.

Thanks!

Questions?