CheckSmart: Cheque Image Feature Extraction System

Report

Amrit Pandey 50604243

Shivanshu Mishra 50605667

1) Project Overview:

This project, developed by Shivanshu Mishra and Amrit Pandey, focuses on building an automated system capable of extracting key information from images of Indian cheques. The primary goal is to process either a single cheque image or a bulk collection of images and extract key fields—such as the payee's name, the date of issuance, the amount in digits, and the amount in words. Instead of detecting these fields dynamically, the system uses manually defined crop coordinates to extract specific regions from each cheque image. Trained OCR models are then applied to these cropped regions to perform handwritten text recognition. The extracted information is structured and saved in machine-readable formats such as JSON or CSV, enabling efficient downstream processing, automated data entry, and digital record-keeping.

Extracting information from documents, particularly those containing handwritten text and varying layouts (even within a standard format like a cheque), is a significant challenge in computer vision and natural language processing. The state of the art in this domain involves leveraging deep learning models for both object detection (to locate fields) and text recognition (OCR), often combining these capabilities in end-to-end document understanding models or using modular pipelines. While commercial solutions exist, developing robust systems for specific document types and regional variations (like Indian cheques with diverse handwriting styles) remains an active area of research and development.

Inputs: The system accepts digital images of Indian cheques. These can be processed individually or as a batch. The dataset used for development and evaluation consisted of 3600 manually curated images of Indian cheques.

Outputs: The system outputs structured data containing the extracted features for each processed cheque. The output format can be either a **JSON** object (per cheque or consolidated) or a **CSV** file, with fields corresponding to Payee, Date, Amount (Digits), and Amount (Words).

1.1) Summary of Contributions:

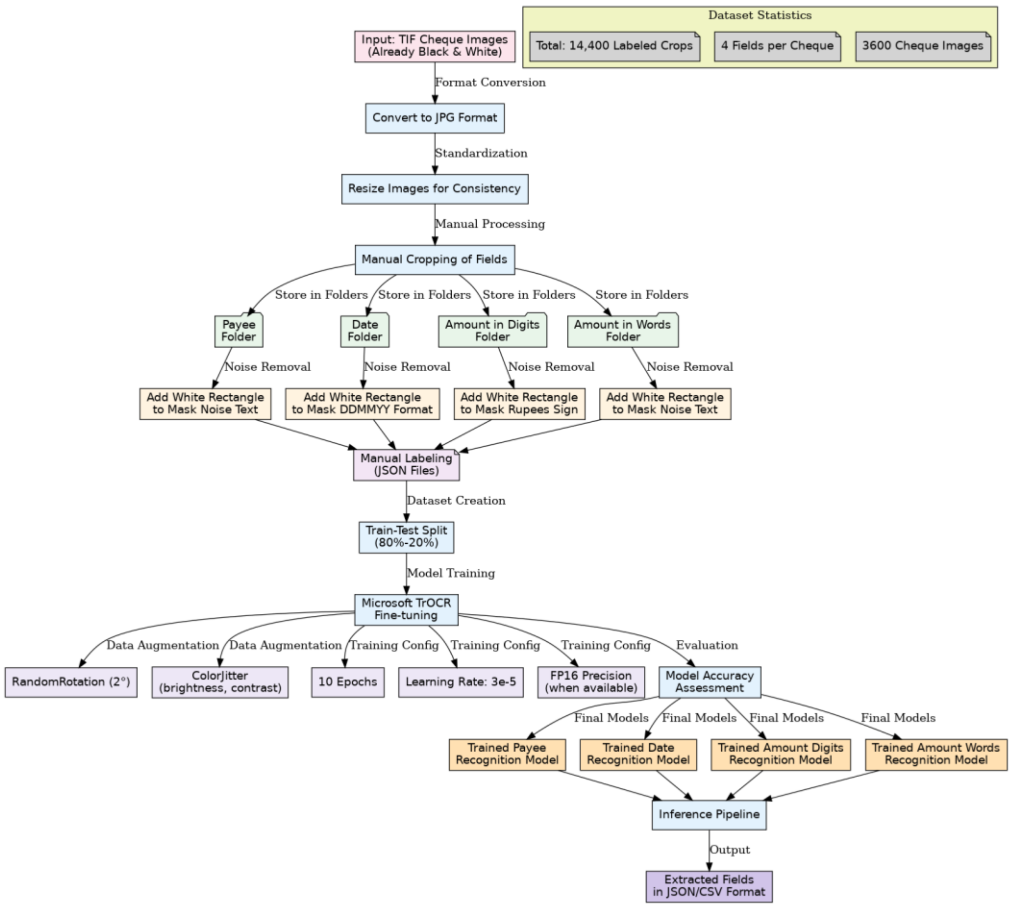
Shivanshu Mishra

- Responsible for labeling the *Payee* and *Date* fields from cheque images.
- Collaborated on the preprocessing pipeline, including image resizing, cropping, and noise removal.
- Experimented with various training parameters for fine-tuning the TrOCR models.
- Deployed the trained OCR models on a Google Cloud Platform Google Cloud Platform (GCP) instance for backend inference.

Amrit Pandey

- Labeled the *Amount in Words* and *Amount in Digits* fields from the dataset.
- Co-developed the preprocessing workflow alongside Shivanshu, ensuring consistency across all fields.
- Tuned model training parameters in parallel with Shivanshu for improved performance.
- Designed and implemented the **backend and frontend** components of the application for end-to-end integration.

2)Approach:


Dataset Statistics
Total: 14,400 Labeled Crops
4 Fields per Cheque
3600 Cheque Images
Input: TIF Cheque Images (Already Black & White)
Format Conversion
Convert to JPG Format
Standardization
Resize Images for Consistency
Manual Processing
Manual Cropping of Fields
Store in Folders
Store in Folders
Store in Folders
Store in Folders
Payee Folder
Date Folder
Amount in Digits Folder
Amount in Words Folder
Noise Removal
Noise Removal
Noise Removal
Noise Removal
Add White Rectangle to Mask Noise Text
Add White Rectangle to Mask DDMMYY Format
Add White Rectangle to Mask Rupees Sign
Add White Rectangle to Mask Noise Text
Manual Labeling (JSON Files)
Dataset Creation
Train-Test Split (80%-20%)
Model Training
Microsoft TrOCR Fine-tuning
Data Augmentation
Data Augmentation
Training Config
Training Config
Training Config
Evaluation
RandomRotation (2°)
ColorJitter (brightness, contrast)
10 Epochs
Learning Rate: 3e-5
FP16 Precision (when available)
Model Accuracy Assessment
Final Models
Final Models
Final Models
Final Models
Trained Payee Recognition Model
Trained Date Recognition Model
Trained Amount Digits Recognition Model
Trained Amount Words Recognition Model
Inference Pipeline
Output
Extracted Fields in JSON/CSV Format

The pipeline illustrates the end-to-end process of building the cheque field extraction system, starting from **TIF cheque images** and ending with **JSON/CSV outputs**. It includes steps such as **image preprocessing, manual cropping, labeling, data augmentation, model training using TrOCR**, and final deployment. Each of the four cheque fields—*Payee*, *Date*, *Amount in Digits*, and *Amount in Words*—was handled with a dedicated model to ensure field-specific accuracy.

2.1) Algorithms Used (Focus on OCR as detailed by user):

We experimented with several approaches for the crucial text recognition step:

- **Donut (Document Understanding Transformer):** We initially attempted to use Donut (microsoft/donut-base), an end-to-end document understanding model, with the goal of directly extracting structured information from the full cheque image. Donut is designed for visual document understanding and generating structured outputs based on document layout and content.
  - *Why we tried it:* Its end-to-end nature seemed promising for directly mapping an image to a JSON output.
  - *Reasons for not adopting:* We encountered significant practical challenges. Training and inference required substantial computational resources (an A100 GPU was found almost mandatory for effective use). More critically, it did not perform well on our dataset, struggling particularly with Indian handwriting, and required extensive, potentially bounding-box-annotated fine-tuning data that was beyond our resources. It failed to reliably extract even basic characters from handwritten fields.
- **Tesseract OCR:** We also evaluated Tesseract, a widely used open-source OCR engine.
  - *Why we tried it:* Tesseract is known for its ease of use and speed, especially on printed text.

    - *Reasons for not adopting:* While functional for printed elements, Tesseract showed very poor accuracy when faced with the variability and nuances of Indian handwriting styles present in our dataset. It did not generalize effectively across different handwriting samples.

- **Final Model Chosen: TrOCR (Transformer-based Optical Character Recognition)**
    - The model selected for the text recognition task was microsoft/trocr-base-handwritten. This model is a Transformer-based encoder-decoder sequence-to-sequence model specifically pretrained for handwriting recognition.
    - *Architecture:* It utilizes a Vision Transformer (ViT) as an encoder to process the input image (or image crop) and extract visual features. A standard Transformer decoder then takes these visual features and auto-regressively generates the output text sequence character by character or token by token.
    - *Why it was effective:* Its pretraining on large handwritten datasets (like the IAM dataset) makes it particularly well-suited for recognizing handwritten text. The Transformer architecture is state-of-the-art for sequence-to-sequence tasks and proved highly effective at mapping image features to text sequences. We found that fine-tuning this model on our specific dataset of Indian cheque handwriting crops led to fast convergence and achieved high validation accuracy compared to the other models tested.

<u>2.2) Original vs Referenced Algorithm Components:</u>

Algorithms and Resources

Here's a distinction between the algorithmic components you implemented/adapted versus those leveraged from external resources:

1. Aspects Coded/Adapted on Our Own:

- **Image Format Conversion & Resizing:** Scripts to convert .tif files to .jpg and resize them to a consistent dimension.
- **Noise Masking Script:** Code to apply white rectangular masks over specific, noisy regions (like "Rs.", "ddmmyy") within the cropped image sections. This involved using manually predetermined coordinates for each section type.
- **Data Loading and Filtering:** Python code to load image paths and corresponding labels from our manually created JSON files, including logic to filter out entries marked as "null" or where image files might be missing.
- **Dataset Preparation:** Code utilizing the datasets library to structure our image paths and labels into Dataset objects suitable for the transformers library.
- **Preprocessing Function (preprocess):** Custom function to load each image, apply augmentations (torchvision.transforms), and use the TrOCRProcessor to convert images and text labels into the required tensor formats (pixel_values, labels) for the model.
- **Custom Collation Function (collate_fn):** A function specifically designed to handle batching of the preprocessed data, ensuring proper padding of label sequences for training the sequence-to-sequence model.
- **Training Orchestration Script:** The main Python script that integrates all components: setting up paths, loading data, splitting data, initializing the model and processor, defining training arguments (Seq2SeqTrainingArguments), configuring and running the Seq2SeqTrainer, saving the trained model, and implementing the evaluation loop.
- **Evaluation Script:** Code to iterate through the validation (and training) dataset, generate predictions using the trained model (model.generate), decode the predictions, compare them against the ground truth labels, and calculate accuracy.

Components Used from External Resources

1. **TrOCR Model Architecture**:
   - The base model "microsoft/trocr-base-handwritten" is from Microsoft Research and Hugging Face
   - Citation: Microsoft Research. (2021). TrOCR: Transformer-based Optical Character Recognition. https://huggingface.co/microsoft/trocr-base-handwritten
2. **Training Framework**:
   - The Seq2SeqTrainer and related training components from Hugging Face's Transformers library
   - Citation: Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38-45).
3. **Data Augmentation Techniques**:
   - Standard PyTorch transforms (RandomRotation, ColorJitter)
   - Citation: Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32, 8026-8037.
4. **Dataset Handling**:
   - The Dataset class from Hugging Face's datasets library
   - Citation: Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., ... & Wolf, T. (2021). Datasets: A Community Library for Natural Language Processing. arXiv preprint arXiv:2109.02846.
5. **Train-Test Split**:
   - The train_test_split function from scikit-learn
   - Citation: Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

2.2) Novel Aspects of Our Implementation:

Our implementation shows innovation particularly in:

1. **Field-Specific Processing**: Our approach to training separate models for each field type (payee, date, amount in digits, amount in words) is a thoughtful design choice that allows for specialized recognition.
2. **Noise Removal Technique**: The application of white rectangles at specific coordinates to mask noise text is a practical and effective approach that doesn't require complex image processing.
3. **Manual Dataset Creation**: The substantial effort in manually creating a dataset of 14,400 labeled crops demonstrates a commitment to quality data, which is often more important than algorithm sophistication.

Standard Practices We Followed

1. **Data Augmentation**: Using rotation and color jitter for augmentation is a standard practice in OCR tasks to improve model robustness.

2. **Hyperparameter Selection**: Your choices for learning rate (3e-5), batch size (32), and training epochs (8-10) align with common practices for fine-tuning transformer models.
3. **Evaluation Methodology**: The exact match accuracy metric is a standard approach for OCR evaluation.

Our implementation effectively combines standard deep learning practices with custom preprocessing and dataset creation techniques tailored specifically for Indian cheque processing. The most significant original contribution appears to be in the dataset creation and field-specific processing approach, while leveraging established models and training methodologies for the machine learning components.

3) Experimental Protocol

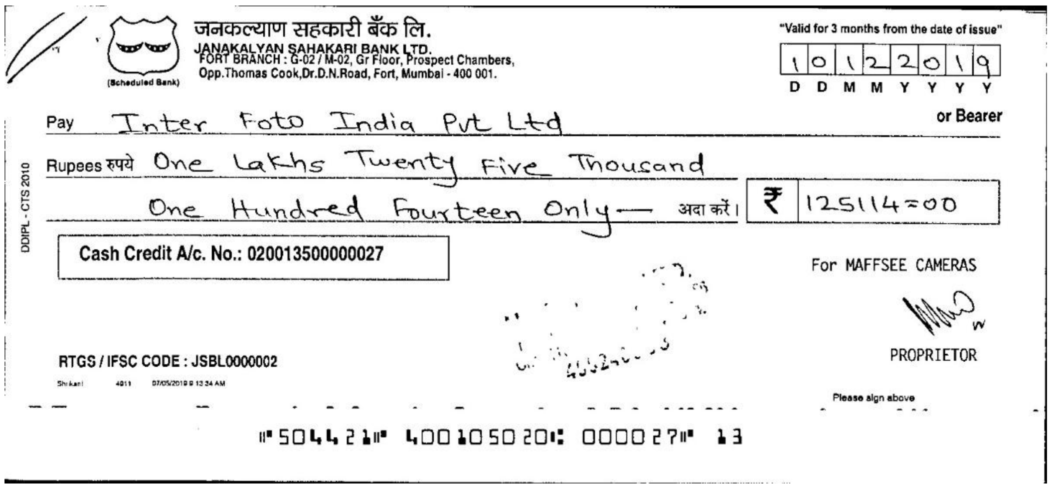
- **Dataset Summary:**
    - **Source:** A collection of ~3600 cheque images, initially in .tif format (black and white).
    - **Preprocessing Steps:**
        1. Converted images from .tif to .jpg format.
        2. Resized all images to a uniform size.
        3. Manually cropped each image into four distinct sections: 'Payee', 'Amount in Digits', 'Amount in Words', and 'Date'. Cropped images were stored in separate folders corresponding to these sections.
        4. Applied white rectangular masks to specific coordinates within the cropped images to obscure recurring noise patterns (e.g., pre-printed "Rs.", "DD/MM/YYYY" text). Coordinates were determined manually for each section type.
    - **Labeling:** Manually transcribed the text content for each of the four cropped sections for all 3600 images. These transcriptions were stored in separate JSON files (e.g., payee_labels.json), linking image filenames to their corresponding text labels. Entries with "null" labels were filtered out during data loading.
    - **Splitting:** The dataset for each section was split into training (80%) and validation (20%) sets using a fixed random state for reproducibility.

3.1) Evaluation Methodology and Compute Resources

• Evaluation Method:

- **Metric:** *Accuracy* — calculated as the percentage of validation samples where the model's predicted text exactly matched the ground truth label. The comparison was done after trimming whitespace and ignoring case differences.
- **Procedure:**
    - The model was evaluated on the remaining **20% validation set**, which was held out during training.
    - For each image in this set, the fine-tuned **TrOCR model** generated predictions using **beam search**.
    - Predictions were decoded using the processor's tokenizer and compared against manually created labels.
    - The total number of exact matches was divided by the total number of validation samples to compute the final accuracy.

3.2) Case Analysis: Model Limitation in Prediction


जनकल्याण सहकारी बँक लि.
JANAKALYAN SAHAKARI BANK LTD.
FORT BRANCH : G-02 / M-02, Gr Floor, Prospect Chambers,
Opp.Thomas Cook,Dr.D.N.Road, Fort, Mumbai - 400 001.
(Scheduled Bank)

"Valid for 3 months from the date of issue"
1 0 1 2 2 0 1 9
D D M M Y Y Y Y

Pay Inter Foto India Pvt Ltd or Bearer
Rupees रुपये One Lakhs Twenty Five Thousand
One Hundred Fourteen Only— अदा करें। ₹ 125114=00

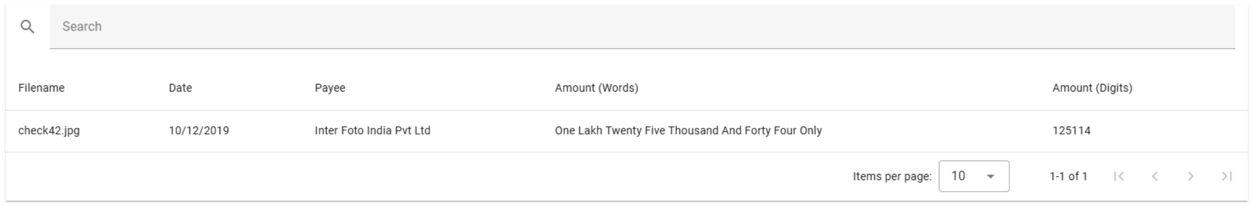
DDIPL - CTS 2010

Cash Credit A/c. No.: 020013500000027

For MAFFSEE CAMERAS
[illegible]
PROPRIETOR

RTGS / IFSC CODE : JSBL0000002
Shikari 4011 07/05/2019 8 13 24 AM
Please sign above

⑈504421⑈ 400105020⑆ 000027⑈ 13


Search

Filename	Date	Payee	Amount (Words)	Amount (Digits)
check42.jpg	10/12/2019	Inter Foto India Pvt Ltd	One Lakh Twenty Five Thousand And Forty Four Only	125114

Items per page: 10 1-1 of 1

In one example (check42.jpg), the model correctly predicted the *Date*, *Payee*, and *Amount (Digits)* fields but made an error in the *Amount (Words)* field. While the actual cheque stated "One Lakh Twenty Five Thousand One Hundred Fourteen Only," the model incorrectly predicted "One Lakh Twenty Five Thousand And Forty Four Only." This reflects a limitation in handling long, handwritten number strings, especially when handwriting is uneven or includes minor variations. It emphasizes the need for more robust post-processing or contextual correction to improve accuracy in complex fields.

4) Algorithm Analysis: Advantages and Limitations:

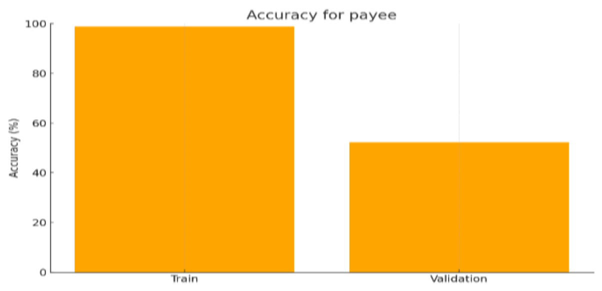
The project employed the **TrOCR model (Transformer-based OCR)** from Microsoft, which combines a Vision Transformer (ViT) encoder with an autoregressive text decoder. This architecture proved highly effective for **structured handwritten text extraction**. Key advantages of using TrOCR include:

- **State-of-the-art accuracy** on clean, well-formatted input (e.g., *amount digits* field reached 92.5% validation accuracy).
- **End-to-end architecture**, eliminating the need for separate feature engineering or segmentation pipelines.
- **Transfer learning** with pre-trained weights allowed fast convergence and high performance even with limited domain-specific data.

However, several **limitations** were observed:

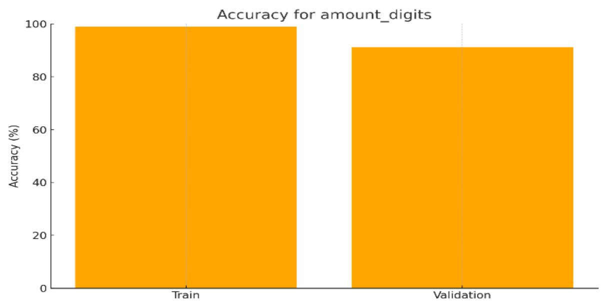
- The model exhibited **overfitting on fields like Payee**, where high training accuracy (98.8%) contrasted with low validation accuracy (51.2%). This is likely due to high variability in handwritten names and cursive scripts.
- **Sensitivity to noise and artifacts** such as background stamps, logos, and faded ink resulted in degraded performance on complex inputs.
- The model struggled with **poorly scanned or tilted images**, especially where character spacing was irregular or cropped.

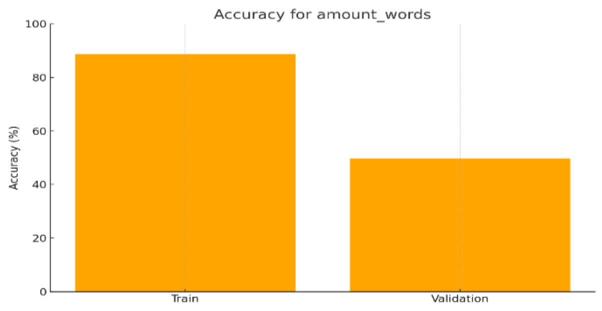
5) Observation:



The payee model showed high training accuracy (98.82%) but low validation accuracy (52.28%), indicating overfitting. This was due to the highly skewed dataset, where a few payee names like "Lic Of India" dominated the distribution. The model struggled to generalize to less frequent payees, which limited its validation performance. Addressing class imbalance is crucial for improvement.

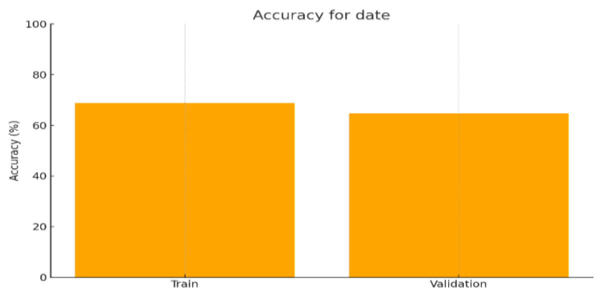
With training accuracy at 99.06% and validation at 91.09%, the amount digits model performed best. The data had a balanced numeric distribution, enabling the model to learn effectively across different values. The minimal accuracy gap suggests good generalization and robustness, making this model highly reliable.

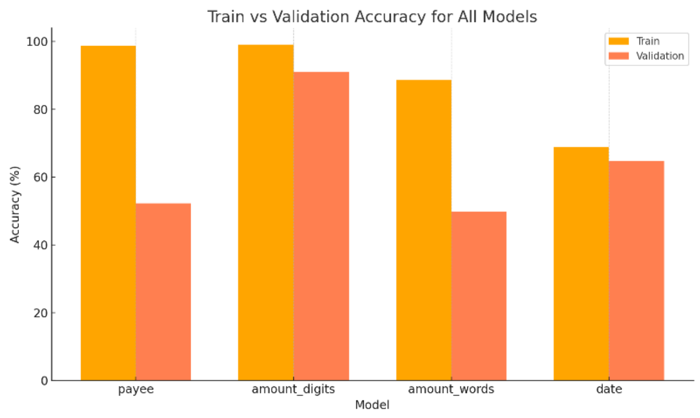




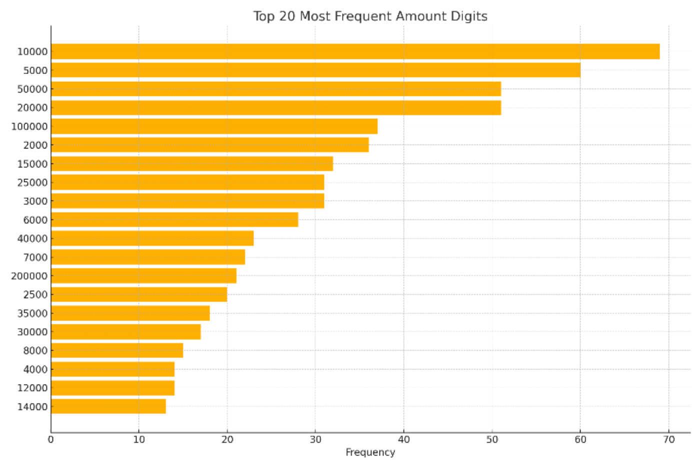
This model achieved 88.60% training accuracy and only 49.78% validation accuracy. The variability in textual representations of amounts led to inconsistent learning. The model likely overfits on common phrases and failed to generalize, showing a need for better text normalization and consistent formatting.

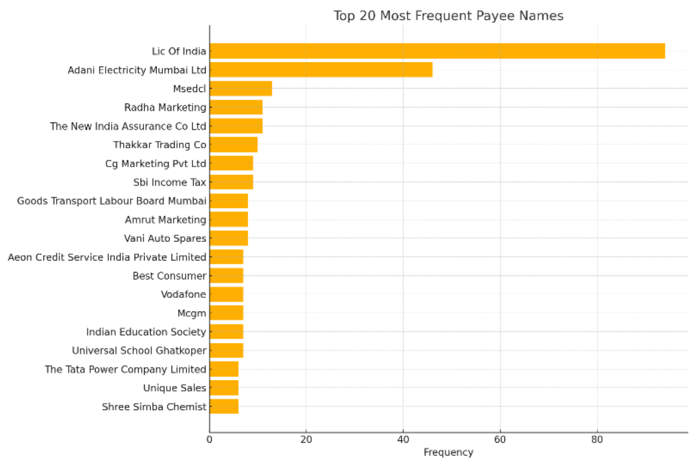
The date model performed moderately, with training accuracy at 68.85% and validation at 64.77%. Its balanced performance suggests minimal overfitting. However, inconsistency in date formats reduced accuracy. Standardizing dates could significantly improve recognition.





Among all models, the amount digits model performed the best, showing high and consistent accuracy between training and validation. The payee and amount words models exhibited significant overfitting due to skewed and inconsistent data. The date model had balanced, though moderate, accuracy likely impacted by format variations. The comparison graph highlights the importance of data balance and consistency for generalization. These observations emphasize that data quality directly influences model performance across all fields.





The performance disparity between the payee and amount_digits models can be attributed to the underlying data distribution. The payee model yielded a low validation accuracy of 52.28% despite high training accuracy, primarily due to a highly skewed dataset where a few payee names like "Lic of India" and "Adani Electricity Mumbai Ltd" dominated the samples, causing the model to overfit and fail to generalize to less frequent payees. In contrast, the amount_digits model achieved a significantly higher validation accuracy of 91.09%, benefiting from a more balanced distribution of numerical values. This diverse representation in the data allowed the model to learn more generalizable patterns, leading to better real-world performance.

The low accuracy is partly due to the evaluation function being too strict, as it compares predicted and ground truth strings for an exact match without accounting for minor OCR-related variations like spacing or missing words. This is unrealistic for handwritten text, where slight deviations are common and shouldn't be penalized as incorrect.

5) Discussion and Lessons Learned

This project provided valuable hands-on experience in applying transformer-based models for document OCR tasks. Key takeaways include:

- **Fine-tuning pre-trained models** like TrOCR are effective but must be complemented with domain-specific cleaning, augmentation, and validation.
- Structured datasets and **consistent labeling** are essential for generalization.
- Evaluation should go beyond accuracy alone to include **editing distance**, **character-level metrics**, or **confidence scoring** for better insights into real-world performance.

From this project, we gained deeper insights into **OCR pipelines**, **model training workflows**, and the **challenges of handwritten document digitization**. This experience will be invaluable in future AI roles, especially in document automation, financial tech, and NLP-driven applications.

References:

- **[1] M. Li, Y. Zhang, Y. Liu, L. Cui, and F. Wei, "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," arXiv preprint arXiv:2109.10282, 2021.**
- **[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.**