Document Classification and Information Extraction with Deep Learning techniques

Gunti Lahari
IIT Hyderabad
ai22btech11008@iith.ac.in

J Hima Chandh
IIT Hyderabad
ai22btech11009@iith.ac.in

S Divija
IIT Hyderabad
ai22btech11026@iith.ac.in

K Anuraga Chandan
IIT Hyderabad
ai22btech11011@iith.ac.in

***Abstract*—Document images are classified based on structure similarity. Instead of hand – crafted features we take CNN which is end to end Learning i.e., it learns the features by itself and also it learns hierarchical nature of document.**

***Keywords*—Classification, Information Extraction, CNN (Convolutional Neural Network), YOLO-v3 , VGG-19, BRNN (Bidirectional Recurrent Neural Network), GRU (Gated Recurrent Units),Bidirectional GRU ,Attention,Data Augmentation,Padding,ReLU,Dropout,Cross Entropy Loss,momentum, SGD**

I. Introduction

There are two approaches 1. text-content based 2. Document structure based. In this we use Document structure based approach. CNN shares weights among neurons in the same layer. Due to inductive biases it is good at finding spatially local correlation by enforcing a local connectivity between adjacent layers with multiple layers CNNs learn the features with tolerance to translation equivariance, and by sharing weights it captures repeating patterns efficiently. We use dropout to prevent overfitting.

II. Our Understanding on Some Research Papers

From [2] the entire workflow moves as follows:

1) *Data preprocessing*
   - Text preprocessing techniques (Tokenization, phrasing etc.) are applied such that they can be used for classification.
   - Compound images are converted into single images using pre-trained CNN model (YOLO-v3 (Fig. 1)).
   - Document Association Network Construction: Constructs a network between related/similar documents.
2) *Image and text Joint Training*
   - Feature Extraction:
     - Text is converted into numerical representation (using word embeddings, etc.).
     - Images are transformed through CNN (VGG-19 (Fig. 2) with slight modifications — final layer replaced by fully connected, dense, output layers on the top) into feature maps.
   - Joint Training via Neural Networks.
   - Fusing Features.
3) *Document classification*

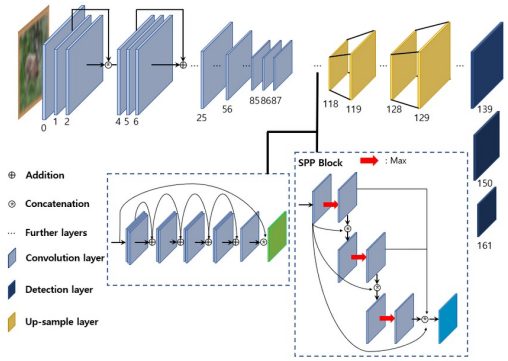

Fig. 1. YOLO-V3 Model [7]

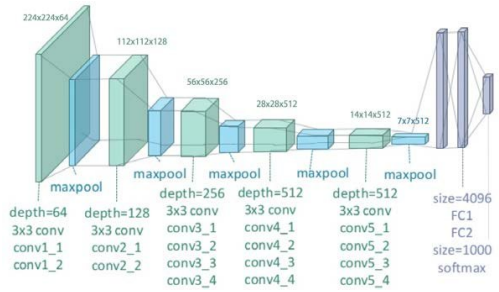

Fig. 2. VGG-19 Model [6]

- Input taken are feature vectors along with document associated network constructed earlier.
- Network fusion learning and classification.

Feature learning was done like text with RNN, image with CNN, network with GNN.

A. Text Learnable Module

The word encoder is built on BRNN (enables utilization of flexible length), GRU to track the state of input sequence.
We already know how the GRU works as said in class. There exists a bidirectional GRU which process data in both directions with forward and backward hidden layers. The sentence encoder uses this representation as input to build sentence level vector similar with document encoder. With the help of attention mechanism to identify the important parts for model.

TABLE VI
SUBCLASS (IPC 4-DIGIT) CLASSIFICATION RESULTS WITH DIFFERENT MODELS

	Model	Top-1 Acc.	Top-5 Acc.	Top-10 Acc.	RAR	Training time (minutes)
(a)	**DeepPatent** [12]	0.615±0.003	0.885±0.010	0.940±0.002	0.233±0.004	9.7
(b)	**LSTM + FHV** [52]	0.605±0.009	0.878±0.007	0.929±0.007	0.221±0.005	14.3
(c)	**LSTM + Word2Vec** [88]	0.591±0.008	0.865±0.004	0.922±0.002	0.206±0.004	13.5
(d)	**GRU + Word2Vec** [53]	0.617±0.008	0.884±0.005	0.934±0.005	0.233±0.004	13.0
(e)	**Fine-tuned BERT** [56]	0.624±0.002	0.891±0.002	0.943±0.002	0.235±0.002	1,290.3
(f)	**TextRCNN** [41]	0.586±0.004	0.860±0.005	0.901±0.005	0.181±0.005	15.7
(g)	**HAN** [42]	0.606±0.001	0.879±0.002	0.932±0.001	0.214±0.001	17.7
(h)	**FastText** [40]	0.484±0.003	0.797±0.002	0.846±0.001	0.100±0.001	3.7
(i)	**Audebert et al.** [65]	0.537±0.002	0.818±0.002	0.875±0.001	0.135±0.002	63.5
(j)	**This work (Text+Image)**	0.636±0.002	0.896±0.002	0.944±0.001	0.236±0.002	77.0
(k)	**This work (Text+Network)**	0.649±0.003	0.908±0.002	0.955±0.003	0.281±0.002	36.3
(l)	**TechDoc (Image+Text+Network)**	**0.655±0.001**	**0.916±0.001**	**0.960±0.003**	**0.292±0.001**	95.3

Fig. 3. Results

With the equations of bidirectional GRU, attention etc., training process is done to derive document vector.

B. Image and text feature fusion learning

In this part, the text feature and image feature are fused and jointly trained via several fully connected layers. There are many levels of classification via some equations (We don't know about them now).
They had considered Cross entropy loss.
The fully connected layer that aims at the lower-level classification task to learn information from the higher-level task through backpropagation
Considering 10 epochs per model their results are as in Figure 3.

III. OUR WORK

Initially, our thought to do project is on information extraction, but as we do more research through documentation, research papers, and GitHub repositories, we expanded the scope to include document classification. The growing importance of document classification across various sectors, such as healthcare, finance, and legal, particularly motivated us in this regard.

We began with document classification of identification cards, as these are critical for verification processes in multiple industries. Our initial focus involves four key document classes:

- Aadhar Card
- Driving License
- IITH College ID Card
- PAN Card

These four classes form the foundation of our classification system. By automating this classification, we like to extend it to other document types such as in Health Care (health records, prescriptions, and medical images), Finance (KYC, Cheque book, Passbook), Passports etc.

A. Data Collection Process

Our initial approach was to search for relevant datasets on platforms like Kaggle and other dataset repositories. However, since many of the domains we are interested in contain sensitive data, it proved challenging to find complete datasets directly from these sources.

We were able to obtain datasets for Aadhar Cards [3] and Australian Driving Licenses [4] from Kaggle. Initially, we had considered including Voter ID cards, but due to the unavailability of a suitable dataset, we decided to shift focus to IITH College ID cards. To gather this data, we circulated a message among friends and collected images of their ID cards.

Obtaining a dataset for PAN Cards was also difficult. Fortunately, we found a GitHub repository [5] that generates synthetic PAN Cards using JSON and Python code. We supplemented this by including images of our own and our friends' PAN Cards to enrich the dataset.

B. Data Preprocessing and Augmentation

In the beginning, we had little understanding of the purpose behind data preprocessing. After watching some videos and reading documentation, we gained insight into its importance.

Initially, our dataset was relatively small, with around 1000 images per class, but only 100 images for the IITH College ID card class. This led us to explore the concept of data augmentation.

To expand our dataset and improve model performance, we applied various augmentation techniques to introduce diversity in the images. These included:

- Resizing all images to a specific size
- Rotating images by a random angle
- Flipping images horizontally or vertically
- Applying zoom in and zoom out transformations
- Adjusting brightness and contrast
- Affine transformations such as shifting or scaling along the axis
- Converting images to tensors and normalizing them using specific mean and standard deviation values

This step is crucial, even with a reasonable dataset size, because real-world test samples can differ significantly, such as partially cropped images, more zoomed-out versions, or flipped images. Without augmentation, our model might struggle to classify these variations correctly. By introducing these transformations, we reduce overfitting and increase the model's ability to generalize, encouraging it to learn the underlying patterns rather than memorizing the data.

We can implement these augmentations using libraries such as imgaug from Python, ImageDataGenerator from Keras, and Transformations from PyTorch, with PyTorch being our tool of choice. We had made the size of each class to 2000.

C. Classification using CNN

Now the preprocessed images can directly be taken as the input to CNN model. The model architecture is as follows:

Network Architecture:

- Convolution layers:
  1) *First Layer*:
     * 16 kernels, each kernel with size 3x3

    * Stride 1, Padding 1
    * ReLU activation
    * Max pooling with kernel size 2x2
  2) *Second Layer*:
    * 32 kernels, each kernel with size 3x3
    * Stride 1, Padding 1
    * ReLU activation
    * Max pooling with kernel size 2x2
  3) *Third Layer*:
    * 48 kernels, each kernel with size 3x3
    * Stride 1, Padding 1
    * ReLU activation
    * Max pooling with kernel size 2x2
- For MLP input, we need to flatten the feature maps of the last convolutional layer.
- MLP (Multi-Layer Perceptron) layers:
  1) *First Layer*:
    * 120 neurons for output
    * ReLU activation
    * Dropout layer with probability 0.7
  2) *Second Layer*:
    * 60 neurons for output
    * ReLU activation
    * Dropout layer with probability 0.7
  3) *Third Layer*:
    * 4 neurons for output
    * ReLU activation
    * Dropout layer with probability 0.7
- We apply the softmax function to get the label of the input image.

Training:

- We use Cross Entropy loss, as the task is Classification.
- For optimization, we use SGD (learning rate = 0.001, momentum = 0.9) with Regularization (decay = 0.001).

IV. Results

Fig. 4. Results obtained by using CNN model

We have trained our model using 25 epochs and the obtained loss 0.1538 and with accuracy 94.17%.

References

[1] Le Kang, Jayant Kumar, Peng Ye, Yi Li, David Doermann University of Maryland, College Park, MD "Convolutional Neural Networks for Document Image Classification" in 2024 22nd International Conference on Pattern Recognition.

[2] Shou Jiang, Jie Hu, Christopher L.Magee, Jianxi Luo "Deep Learning for Technical Document Classification" in February 2022 IEEE Transactions on Engineering Management published online.

[3] https://www.kaggle.com/datasets/monsterslayer/aadhar-all-data

[4] https://www.kaggle.com/datasets/turabbajeer/forged-characters-detection-on-driving-licence

[5] https://github.com/shobhitchittora/PAN-dataset

[6] https://www.kaggle.com/code/abanobmorgan/vgg19-classifier

[7] https://www.researchgate.net/figure/The-network-architecture-a-YOLOv3-b-YOLOv4_fig2_348825339