DoodleVision: Transforming Hand-Drawn Sketches into Digital Reality

Aryan Agarwal
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(24110012) aryana24@iitk.ac.in

Himalaya Kaushik
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(241110029) himalayak24@iitk.ac.in

Keshav Banka
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(241110032) bkeshav24@iitk.ac.in

Yuvraj Raghuvanshi
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(241110084) yuvrajpr24@iitk.ac.in

Tsewang Chukey
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(241110092) tsewang24@iitk.ac.in

Tsewang Namgail
Dept. of Computer Science and Engg.
Indian Institute of Technology, Kanpur
India
(241110093) tsewangn24@iitk.ac.in

***Abstract*—In era of digital transformation, many techniques have been developed to enhance user experience in field of image recognition for many use cases such as education industry, designing and software industry and persons-with-disability assistance tools etc. Our project aims to compare different techniques in field of Deep Learning using two types of data- Strokes and Images, for task of Hand Drawn Sketch recognitions at relatively optimal inference speed for use cases mentioned before and simultaneously compare their performance for industrial usage including on smaller devices. We also propose a novel approach of hybrid model designed to have parallel training on both strokes and image data, which outperforms all models studied in this report.**

***Index Terms*—Hand Drawn Sketch Recognition, Vision Transformers, LSTM, Deep Learning, InceptionNet, ConvLSTM, Convolutional Neural Network, MobileNet , VGG, Resnet, Hybrid Models**

I. Introduction

In field of Deep Learning many advances have been made to digitalize the human based sketches for better user experience and ease of working [1] [2] [3] [4]. To understand the working and picking the right approach for a given problem at hand hence becomes are necessary challenge that we aim to solve in our project. We focused on developing a system where user can draw objects by hand, we capture its strokes and image data that is fed into one of the deep learning models to predict in real time what the object is or which of the class it belongs to in our dataset. For this we made use of Google Creative Labs dataset for sketch drawings. It has 345 classes with millions of sample image or stroke data points. We conducted comparative analysis across techniques and ablation studies to understand what all different deep architectures have as weaknesses and strengths in a given dataset scenario and compute limitations on the training side. By this we ensure a good accuracy prediction at real time with good speed so that it can be adopted in to smaller devices as well for further industry use cases such as mobile apps for persons with disabilities to assist them with object drawing recognition and integrate in further smart home devices systems to help them, or creative design tools for educational industry or design industry for object recognition etc as mentioned in our presentation slides.

II. Related Work

Prior works explore LSTM [5], ConvLSTM [6] and Transformer Based [7] approaches for modeling sequential strokes data and spatiotemporal features, enhancing real-time object prediction. They also covered depth exploration with ConvLSTM which is an use case specific ability of these models. Sketchformer [7] introduced transformer based models that makes use of attention mechanism to discriminate between strokes vectors for different object and able to understand much more complexity of dataset at hand. CNN based models [8] [9] and pretrained models have also been explored like VGG, Resnet and MobileNet [10] [11] [12] implementing concepts of skip connections and strong feature extraction capabilities of Convolution layers which is pretrained on larger datasets with higher computation power that we can fine tune for our dataset with limited training resources. Hence with these works in place, we did detailed study including ablation cases of these techniques into our models for both strokes vector and raster image data to understand model sensitivity to input type, depth and attention while optimizing for overall accuracy, inference and training speed, and arrived at overall analysis of each method compared to results from these prior works and across techniques for better selection in future.

III. Dataset

In each of the techniques, We have used Google Creative Lab's Dataset which has over 50 million acorss 345 categories for Strokes and Image data both. Its sketch

doodles data they have collected and is open source. https://github.com/googlecreativelab/quickdraw-dataset

IV. Methodology

We have multiple approaches listed below that we implemented for solving the problem at hand. Experimental Setup and Methodology details are listed for each technique for better clarity.

A. LSTM

In LSTM based approach, each is a sequence of pen movements having x , y coordinate and binary value of pen up or down.

1) Experimental Setup and Implementation Details

In training we send these strokes data in one-by-one manner to inculcate the timestep logic with RNN based models excel at working with, and understand the different strokes pattern, loops, angles or pauses etc and then overall output is predicted using these logics. The model layers include this flow: Masking → LSTM → Dense → Dropout → Dense SoftMax.
Challenges included replicating strokes data at inference to match same pattern as training data, handling issues of LSTM that includes capturing long term dependencies if the stroke is very complex. We used 20 thousand samples per class in our dataset for training. For hardware limitations we had to make these decisions. Figure 1 reflects strokes data collected and plotted at inference similar to training data samples. Model Parameters count is 395288.

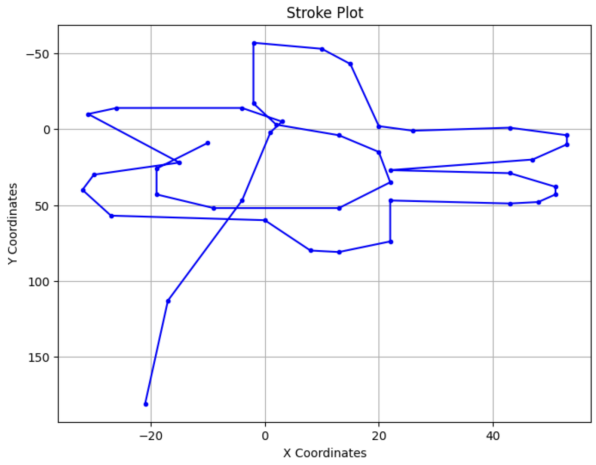

Fig. 1. Strokes Data collected in Inference

2) Results and Analysis

Table 1 reflects the model accuracy attained over test dataset. Figure 2 and 3 reflects training accuracy and loss over training epochs respectively.

B. ConvLSTM

ConvLSTM extends LSTM by adding Convolution operations, allowing it capture spatial features as well along with temporal dependencies. Strokes are processed in similar manner as LSTM.

TABLE I
TEST RESULTS

Method and Data type	Accuracy	Loss (Nats)
LSTM on Strokes Data	81.93%	0.6170

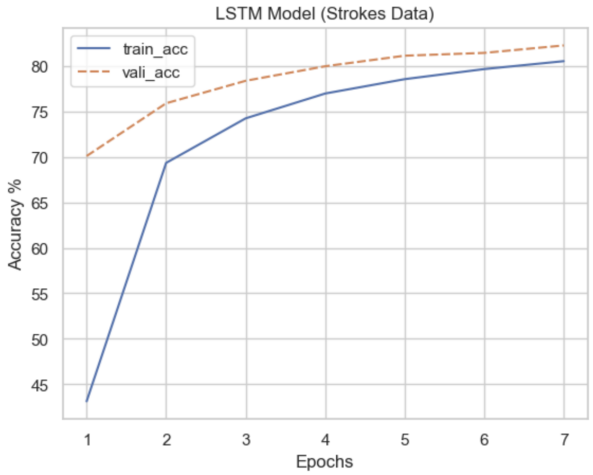

Fig. 2. Accuracy Curves over Training Epochs for LSTM Model

1) Experimental Setup and Implementation Details

Training and Inference becomes slower here, along with number of parameters scaling quite high to 798,904 and almost 27 mins per epoch training. Model layers lfow is Reshape → ConvLSTM → Flatten → Dense → Dropout → Dense Softmax Challenges including managing larger training time, transforming sytroke data into 2D temporal frame for convolution input.

2) Results and Analysis

Table 2 reflects the model accuracy attained over test dataset. Figure 4 and 5 reflects training accuracy and loss over training epochs respectively.
Also, we infer that for such massive scale in parameters, and only minimal improvement over accuracy in test data, LSTM is much better for our dataset case and inference environment. A comparison in model size is shown in Figure 6.

TABLE II
TEST RESULTS

Method and Data type	Accuracy	Loss (Nats)
ConvLSTM on Strokes Data	87.43%	0.5078

C. Vision Transformers for Image Data

Vision Transformers (ViT) applies transformer architecture to image data by making image data into patches, adding positional encoding and treating them as sequences and then

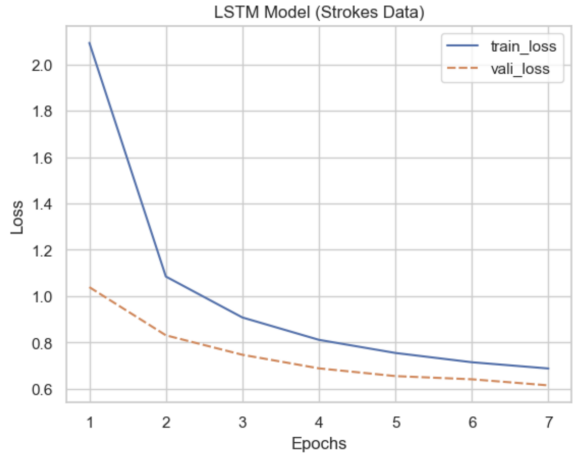

Fig. 3. Loss Curves over Training Epochs for LSTM Model

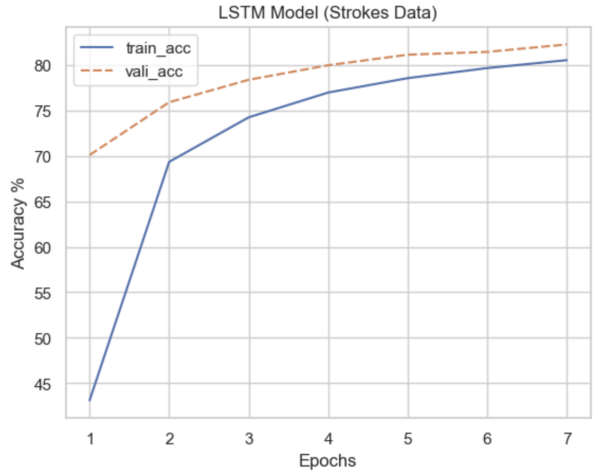

Fig. 4. Accuracy Curves over Training Epochs for ConvLSTM Model

applying attention mechanism on it. This enables global context learning without convolutions!

1) Experimental Setup and Implementation Details

We take grayscale images of size 28 x 28 and split into patches of 4 x4. Then these values are finalized after a bit of hyperparameter tuning. then these patches are linearly embedded into a vector with positional encoding added to it, ehich fed into transformer blocks with attention mechanism. This is flow of layers : Patch $\rightarrow$ Embedding $\rightarrow$ Positional Encoding $\rightarrow$ Transformer Encoder $\times$ N $\rightarrow$ Global Pool $\rightarrow$ MLP $\rightarrow$ SoftMax. We used GeLU activations here after seeing betetr perfomance with it.

2) Results and Analysis

Ofcourse with attention, the number of parameters scaled to almost 10 times than convLSTM, and so did the model size around 30 MB, thus making it difficult for smaller devices. Train accuracy saturated around 82% after 10 epochs, for our configurations which was kept relatively simpler as we had

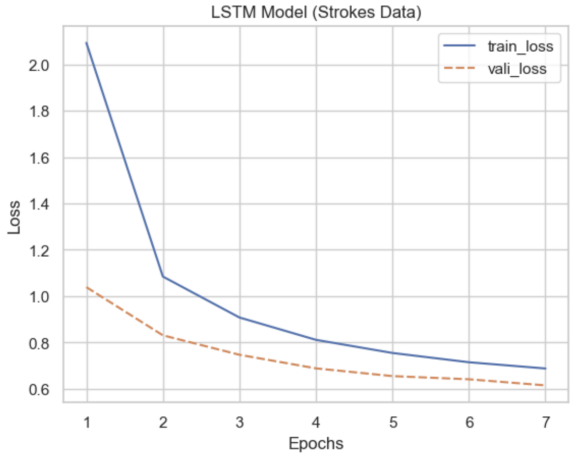

Fig. 5. Loss Curves over Training Epochs for ConvLSTM Model

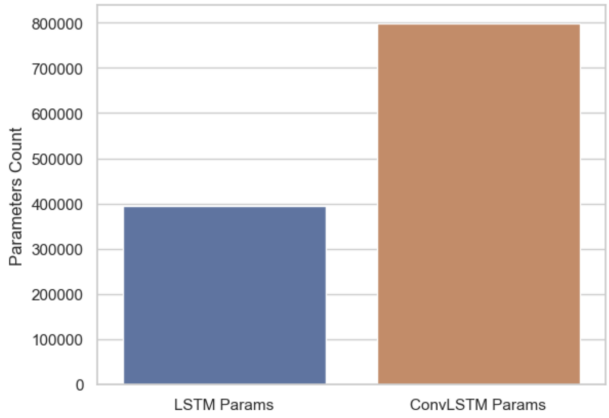

Fig. 6. Model Parameters comparison in LSTM vs ConvLSTM models

hardware limitations and hence we dealt with what we have in most optimal manner. Number of Parameters are 7.93M. Table 3 reflects the model accuracy attained over test dataset. Figure 7 and 8 reflects training accuracy and loss over training epochs respectively.

TABLE III
TEST RESULTS

Method and Data type	Accuracy	Loss (Nats)
Vision Transformer on Image Data	80.00%	0.6960

D. Transformers for Strokes Data

Transformer model supersedes LSTM model to find long range dependencies in the strokes data. This setup captures global dependencies through self attention mechanisms across multiple heads in the layers of model.

1) Experimental Setup and Implementation Details

In this, strokes are processed using self attention mechanism adopting step by step recurrence of a sequence like in text

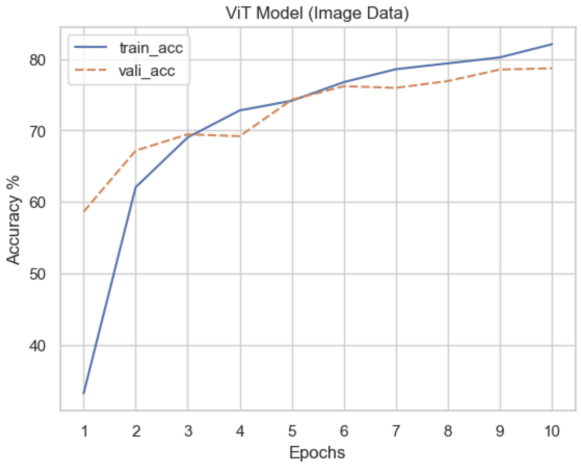

Fig. 7. Accuracy Curves over Training Epochs for ViT (image) Model

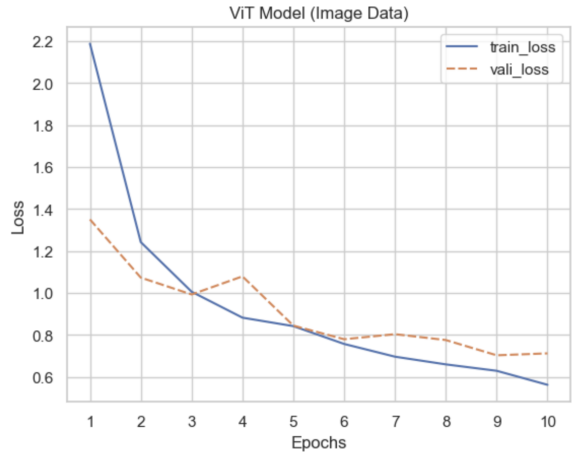

Fig. 8. Loss Curves over Training Epochs for ViT (image) Model

or LSTM based models. This made model capture complex patterns such as loops, pauses, angles etc. The architecture flow is: Positional Encoding → Multi-Head Attention → Layer Normalization → Feed Forward → Dropout → Dense Soft-Max. Major challenges included preserving sequence order or strokes which was handled using positional embeddings.

2) Results and Analysis

We used 20 thousand samples per class in this. Table 4 reflects the results attained. For these larger parameters based models, we had limited resources to train on, hence more future scope of work remains with better resources.

TABLE IV
TEST RESULTS

Method and Data type	Accuracy	Loss (Nats)
Transformer on Strokes Data	74.86%	0.8987

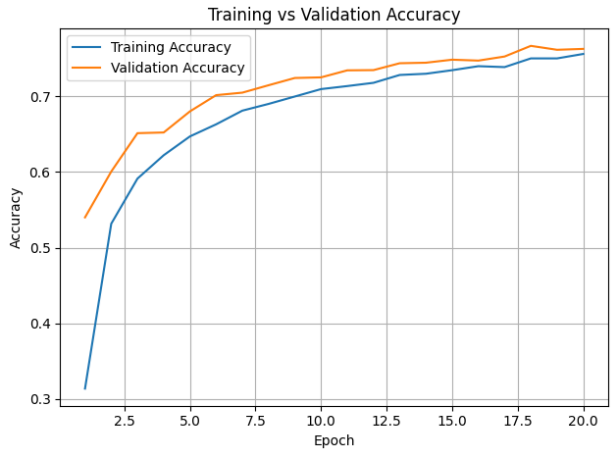

Fig. 9. Accuracy Curves over Training Epochs for Transformer Model

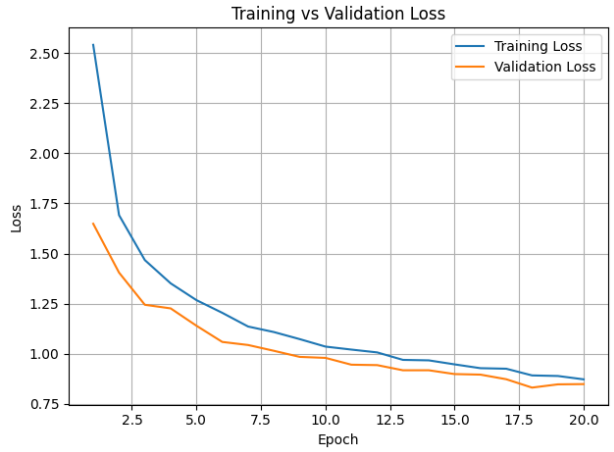

Fig. 10. Loss Curves over Training Epochs for Transformer Model

E. Hybrid Model for Strokes + Image Data

We came up with a novel approach to implement both type of datasets strokes and image to capture strengths from both techniques of sequential stroke based modeling and spatial image based feature extraction. In this we make use of Bidirectional LSTM to capture temporal dependencies in stroke data $(\Delta x, \Delta y, \text{pen state})$ and for image (grayscale) we use CNN based feature extraction layers to extract spatial patterns. These are trained in parallel and concatenated at the end, thus making the approach efficient, scalable and fast at inference making it suitable for small edge devices.

Model Flow:

CNN(Image) → Dense ↔ BiLSTM(Stroke) → Dense → Concat → Dropout → Dense (SoftMax)

1) Experimental Setup and Implementation

Stroke input was of shape (130, 3) and image input was (28, 28, 1), with both normalized before training. The stroke side used a Bidirectional LSTM, while the image path used a CNN architecture. Feature embedding from both branches were joined and passed through a fully connected MLP network.

This dual-input approach allowed the model to learn both features: temporal dynamics from strokes and spatial structure from image pixels.

2) Results and Analysis

Model achieved **93.12%** validation accuracy outperforming all models we explored. This was effective in distinguishing very similar classes like cup and bowl.

Key Insight:

Mixing the temporal data and image data enhanced the class prediction and distinction, thus making model robust for real time uses. Also, the model parameters were comparatively less and parallel branches helped imporve training and inference time.

Model Parameters: ∼0.57 million

Training Details: 20 epochs, batch size optimized for hardware constraints, fused features trained jointly.

TABLE V
TEST RESULTS

Method and Data type	Accuracy	Loss (Nats)
Hybrid model on Strokes+Image Data	93.72%	0.2069

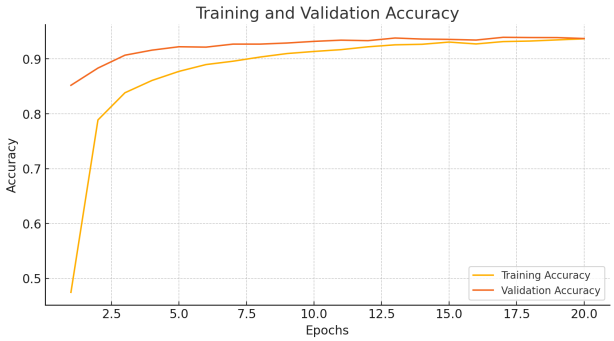

Fig. 11. Accuracy Curves over Training Epochs for Hybrid Model

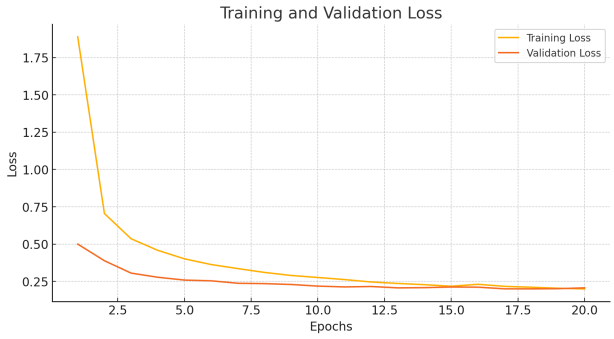

Fig. 12. Loss Curves over Training Epochs for Hybrid Model

F. Convolutional Neural Network

CNNs have been widely used to extract information from images. In our case CNN takes the spatial information from the image using the convolution operations, flattening it to a 1D vector, and feeding it into a dense layer of neural network to output the final prediction.

1) Experimental Setup and Implementation Details

We take 20 thousand images of size 28x28 from each class and split the dataset in 80-10-10 format, normalize it, and feed it through the network which includes: Conv2D (32x3x3) → MaxPooling2D (2x2) → BatchNormalization → Conv2D (64x3x3) → MaxPooling2D (2x2) → BatchNormalization → Flatten → Dropout (0.4) → DenseLayer(128) → Dropout (0.2) → Softmax. The model uses Adam optimizer with learning rate of 0.001 and Sparse Categorical Cross Entropy (SCCE) loss function. We have also used early stopping based on validation accuracy with patience of 5.

2) Abalation Study, Results, and Analysis

For the ablation study we tweak a few major components of our architecture which include architecture depth, number of neurons in dense layer, and dropout percentage. Taking the above defined architecture as base the following modifications are made:

- No Dropout: Both of the dropout layers are reset to 0
- Deeper Conv Layers + Wider Dense: The Conv2D layers are changed from [Conv2D (32x3x3), Conv2D (64x3x3)] to [Conv2D (64x5x5), Conv2D (128x3x3)] and Dense layer is changed from 128 to 256.

TABLE VI
PERFORMANCE COMPARISON

Method	Accuracy	Loss(SCCE)
Base	87.24%	0.48
No Dropout	86.54%	0.53
Deeper & Wider	**89.41%**	**0.4**

During our ablation study of CNNs we have found that without the dropout layer the network quickly plateaus and triggers early stopping leading to far worse test accuracy 13 14. On the other hand with deeper convolutional layers and wider dense layer the network outperforms the base network with the cost of model size and number of trainable parameters 15.

G. Inception V3

To capture multi-scale features and properties, the deep convolutional neural network InceptionV3 uses parallel convolutions of different kernel sizes at each convolutional layer. Compared to the results in the paper *Meta Analysis of Deep Learning Models for Doodle Recognition* (2021), which assessed it on a small set of 25 classes without unfreezing different layers, we used it as a pretrained feature extractor for our task on much more classes.

Our dataset provided Greyscale 28×28 images. They were transformed to three channels and enlarged to 75×75 in order to pass them to Inception V3. For preprocessing, we interpolated to the range $[-1, 1]$. Training took place using

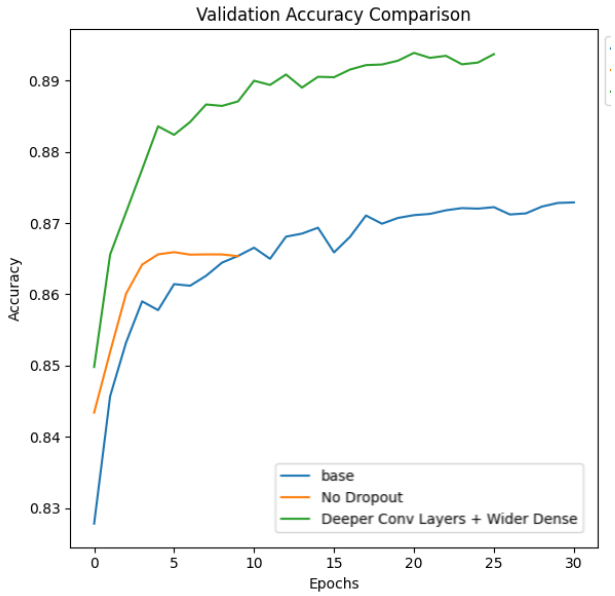

Fig. 13. Accuracy Curves over Training Epochs for CNN Ablation

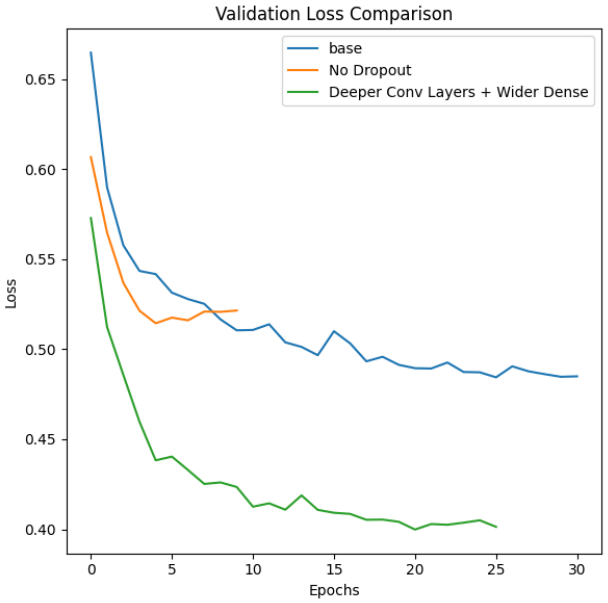

Fig. 14. Loss Curves over Training Epochs for CNN Ablation

the Adam optimiser, ReLU activations for classifiers. 3 Hidden and 1 output layers were added at the end of Inception feature extractor.

1) Experimental Setup and Implementation Details

Inception V3 was used as a feature extractor and a classification layer was added at top of model consisting of fully connected layers with batch normalization and dropouts for regularization. Only selected layers of the base network were unfrozen based on ablation results.

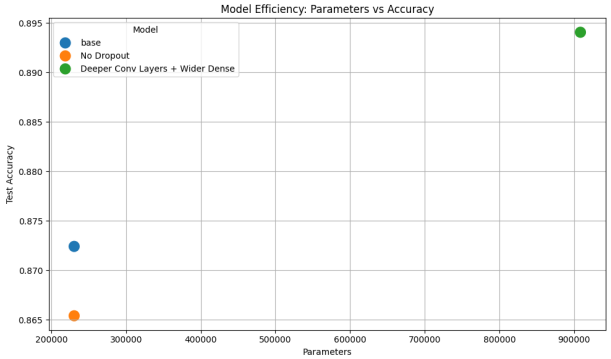

Fig. 15. Parameters vs Accuracy chart for CNN Ablation

2) Ablation Study, Results and Analysis

Progressively unfreezing different layers of the InceptionV3 backbone presented consistent gains in accuracy and reduction in loss. The best performance was achieved when layers from `mixed6` onward were unfreezed and fine-tuned. We found a continuous increase in performance on unfreezing more and more layers. This was observed because fine tuning more layers lead to better feature extraction and capturing of different sizes of features in image. We stopped at `mixed6` because fine-tuning it further defeats the whole idea of using a pre-trained model.

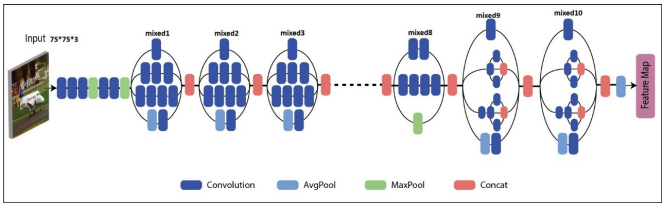

Fig. 16. InceptionV3 model

TABLE VII
PERFORMANCE COMPARISON OF LAYER-WISE FINE-TUNING

Fine-tuned Layers	Accuracy	Loss (nats)
Head Only	70.35%	0.71
+ mixed10	69.54%	0.66
+ mixed9	77.92%	0.62
+ mixed8	81.45%	0.59
+ mixed7	85.03%	0.56
+ mixed6	**86.06%**	**0.5487**

H. Pretrained Model (VGG16)

VGG16, introduced by Simonyan and Zisserman (2014), is a deep convolutional neural network known for its effective

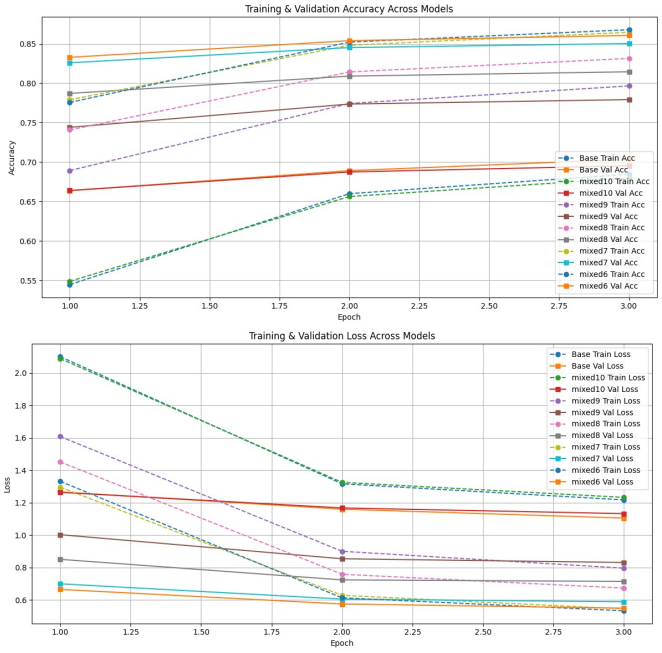

Fig. 17. Ablation study results: Accuracy (top) and loss (bottom) across different unfreezing depths in InceptionV3.

feature extraction via stacked 3×3 convolutions and for its simplicity. Several doodle and sketch recognition tasks used it due to its ability to model hierarchical patterns.

In the paper *"Meta Analysis of Deep Learning Models for Doodle Recognition"(Sethi & Duhan, 2021)*, [11] the authors included various deep learning architectures which includes standard CNNs, RNNs and hybrid models on doodle datasets. Reported accuracies for all these models typically ranged between 70% and 84% while depending on the complexity of the model and training volume data. In contrast, our approach by using a pretrained VGG16 model and by fine tuning, achieved a accuracy of up to 88.10%. This shows a significant performance gain over the models in meta-analysis, hence validating the effectiveness of transfer learning with deep, structured CNNs like VGG16 for abstract classification of sketch. By fine tuning, our model maintained computational efficiency while generalization being improved.

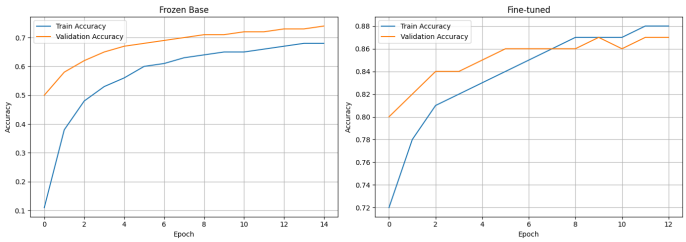

Fig. 18. Training and Validation Accuracy – Frozen Base and Fine-Tuned (VGG16)

1) Experimental Setup and Implementation Details:

As shown in the model architecture Fig. 19 We have used a modified VGG16 architecture which is pretrained on Imagenet, in which initially the convolutional base was frozen to high-level features from the sketches. A custom classification head consisting of Global Average Pooling layer,a dense layer with ReLU activation, followed by a Dropout layer and finally a dense layer with softmax activation function.

During the fine-tuning only the selected block (block5) convolutional layers were unfrozen which allows the model to adapt more effectively to the abstract sketch domain while also preserving the efficiency of the pretrained feature extractor.

We first kept Block 5 of VGG16 model frozen so the features that the model has already learned from the ImageNet dataset could be use by the model helping in avoiding overfitting and making the training more faster. Later on we unfroze this block in order to allow the model to learn features which are specific to our dataset thich further helped in improving the overall accuracy.

Fig. 19. Model Architecture VGG16

2) Ablation Study, Results and Analysis:

As shown in Figure 18 During the ablation study we have found that the frozen base has showed steady improvements,

but after fine-tuning it brought a noticeable gain in validation accuracy and convergence.

In the table VIII *Method A* shows the accuracy and loss of Frozen Base and whereas *Method B* shows the accuracy and loss of fine-tuned (unfrozen) achieving strong performance.

TABLE VIII
PERFORMANCE COMPARISON (VGG16)

Method	Accuracy	Loss (nats)
Method A	68.30%	1.00
Method B	88.10%	0.48

I. Pretrained Model(MobileNetV2)

MobileNetv2 is used for its lightweight architecture that helped us to reduce model parameters and inference time, while delivering good accuracy. It is optimized for mobile and embedded vision applications. It consists of depthwise separable convolution layers and inverted residual blocks as well. These reduce computation effort a lot. Compared to other pretrained models, it gave better performance

As compared to the paper *"Meta Analysis of Deep Learning Models for Doodle Recognition"(Sethi & Duhan, 2021)* [11] our MobileNetV2 based model, pretrained on ImageNet and fine-tuned with a custom classification head has achieved an accuracy of 87.11% that outperforms the upper bound of models evaluated in the paper. This result shows the advantage of using efficient transfer learning architectures like MobileNetV2 which is lightweight with high representational power even on the abstract sketch data.

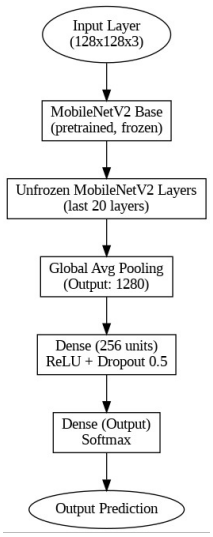

Fig. 20. MobileNetV2

1) Experimental Setup and Implementation Details:

As shown in the architecture figure 20, while adopting the transfer learning setup using pretrained MobileNetV2 on the dataset. We have kept a smaller flowchart to explain model architecture, as listing all layers would be very long and uninterpretable in a figure. All images were initially resized from 28×28 to 128×128 and then converted to RGB format and then on which normalization and data augmentation was applied during training.

MobileNetV2 base, which is pretrained on ImageNet was used with its weight frozen and a custom head consisting of a 'Global Average Pooling' layer, followed by a 256-unit of Dense layer with ReLU activation, Dropout and finally a softmax layer was added. Adam optimizer was used to compile the model and categorical cross entropy loss and trained for 10 epochs using early stopping and learning rate scheduling.

Initially in order to use the pretrained knowledge for basic features we kept all the layers frozen. Later on, we unfroze the last 20 layers so that our model could learn more detailed and specific patterns from our own dataset, which helped in improving the model's accuracy and performance.

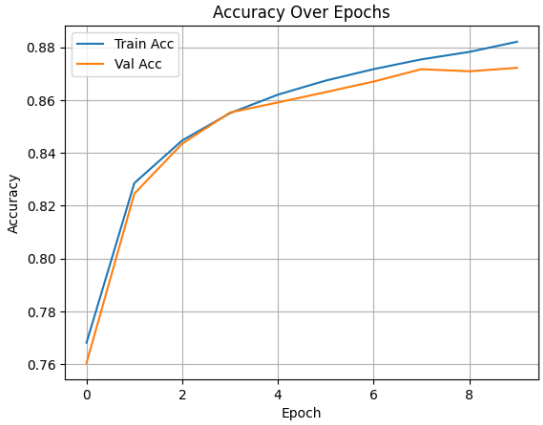

Fig. 21. Training and Validation Accuracy over 10 epochs (MobileNetV2)

2) Ablation Study, Results and Analysis:

As shown in the figure 21, the model initially used frozen base layers in which only the top classification head was trained (Method A). Using the Adam Optimizer with a learning rate of 0.0001, this method achieved an accuracy of 79.15% which is a reasonable baseline for feature extraction.

In contrast, in *Method B* which involved fine-tuning the model by unfreezing the top 20 layers of the MobileNetV2 model and by slightly increasing the learning rate to 0.001. This method significantly improved the model's ability to get adapt to the sketch data, hence resulting in a much higher accuracy of 87.22% and a loss of 0.48 as shown in the below table IX.

These results clearly shows the effectiveness of selective fine-tuning in enhancing the model performance on the abstract input domains like the given sketches.

The model was highly sensitive to learning rate. Once tuned, it achieved strong accuracy with low inference time (16ms/image), demonstrating its effectiveness for deployment on the resource-constrained devices. Its lightweight nature makes it really suitable for real-time or embedded doodle recognition applications.

TABLE IX
PERFORMANCE COMPARISON (MOBILENETV2)

Method	Accuracy	Loss (nats)
Method A	79.15%	0.76
Method B	87.22%	0.48

J. Pretrained Model (ResNet)

ResNet50 is a deep CNN architecture with the concept of residual connections and allows for much deeper networks without the problem of vanishing gradient.

In comparison with the paper [11] we found that our ResNet50 model which was fine-tuned by unfreezing the top 20 layers, achieved an accuracy of 79.49% which falls in the performance range discussed in the meta-analysis paper. We found ResNet to be a competitive choice even for abstract sketch inputs, while offering a good balance when applied with proper fine-tuning.

1) Experimental Setup and Implementation Details

As shown in the architecture diagram 22 the base architecture was initialized with the pretrained ImageNet weights and the images were resized to 128×128 pixels and then normalized to [0, 1]. For classification head, a Global Average Pooling layer was added, followed by a 256-unit Dense layer with ReLU activation, dropout and finally a softmax output layer. We have kept a smaller flowchart to explain model architecture, as listing all layers would be very long and uninterpretable in a figure.

Initially in order to use the pretrained knowledge for basic features we kept all the layers frozen. Later on, we unfroze the last 20 layers so that our model could learn more detailed and specific patterns from our own dataset, which allowed the model to adapt better to our task and in improving the accuracy.

2) Ablation Study, Results and Analysis

We did ablation study for this model as well. Method A in table X refers when only classifier was trained, with frozen base layers. Results are mentioned in the table. To improve, Method B adds fine tuning approach by unfreezing last 20 layers i.e we train those layers parameters. We also reduced learning rate of 0.001 for this method to arrive at better accuracy. details are mentioned in table.

In the figure 23 it shows that after unfreezing the last 20 layers of ResNet50, both the training and validation accuracy shows steady improvement achieving around 80% which indicates good generalization and throughout the fine-tuning process stable learning.

TABLE X
PERFORMANCE COMPARISON

Method	Accuracy	Loss
Method A	48.32%	1.96
Method B	79.49%	0.74

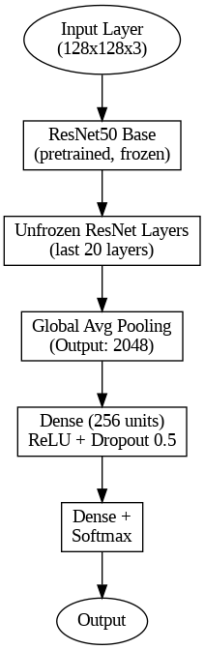

Fig. 22. ResNet50 model

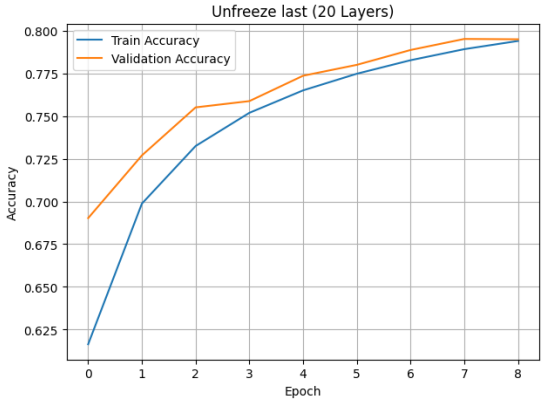

Fig. 23. Training and Validation Accuracy over 9 epochs (ResNet50)

V. Results and Comparison Analysis

Results of individual techniques is mentioned in respective sections. Here we talk about overall comparative analysis.
As seen from Table XI we can understand how ech model is performing on metrics like accuracies and parameter sizes. With these we can come to conclusions as follows-

1) **Our Novel Hybrid Model** that uses Image Spatial information and Strokes sequential patterns enhanced the prediction to an **amazing 92+% accuracy**, thus making it the best model. Its **parallel training architecture** improved overall speed and inference as well. Its small model size makes it **feasible for small edge device deployments** as well.
2) ConvLSTM gives better accuracies but, it comes with heavy tradeoff of number of parameters, hence its not advisable for small models for such less change in

accuracy. Also inference time increases for convLSTM, hence not a suitable choice

3) We can see as we increase model complexity with transformers, the model seems to reach better accuracy in less epochs, but they have humongous model sizes
4) Based on ablation studies, we can conclude that for task specific models, which layers contribute the maximum as seen in CNN and InceptionNet ablation studies
5) the pretrained models comes in handy the most, as they do have sufficient training done in advance with better hardware resources, which we didnt have.
6) VGG is performing well, but MobileNet is more optimal choice due to smaller size relatively and can be efficiently implemented in devices where quick prediction and inference is needed.
7) Basic CNN is most flexible as we can make task specific requirement changes without adding a lot of model complexity (like in transformers) to attain better accuracy and very good inference speed.
8) In all the analysis strokes data handling was more complex to handle and train, which makes sense as well and hence it can be optimally implemented with LSTM based models and enhanced layers mechanism to capture more complex figures in future.

TABLE XI
COMPARATIVE ANALYSIS OF ALL TECHNIQUES

Technique	Inference Time (Avg)	Model Parameters	Accuracy
LSTM	25ms	0.39M	81.93%
ConvLSTM	110ms	0.79M	87.43%
Transformer (Strokes)	80ms	0.62M	76.29%
Vision Transformer (Image)	40ms	7.93M	79.96%
CNN	12ms	0.23M	87.20%
Pretrained: InceptionV3	70ms	0.28M to 15.2M	86.30%
Pretrained: VGG	183ms	14.8M	88.10%
Pretrained: ResNet	72ms	24.12M	79.49%
Pretrained: MobileNet	16ms	2.5M	87.22%
Hybrid Model	**23ms**	**0.57M**	**93.72%**

VI. Inference and Implementation Challenges

1) We developed a real-time interface system with interactive user interface that allows user to draw sketches or objects in air and that drawing is captured using standard webcam. Figure 24 shows an real-time example. We also made a UI where user can draw using mouse pointer. In both UI, we are able to capture image data of object drawn along with strokes patterns etc, and this data is used for prediction objection from our respective models for corresponding data type. The inference speed is very good, and with smaller models like LSTM or Mobilenet , we can extend this UI to **smaller devices as well thus reducing model footprint and ease of scalability!**
2) This method enables a touch less , intuitive experience for user, **particularly beneficial for assistive technologies helping persons with disabilities, allowing them to trigger actions by just certain gestures/drawings in air.**

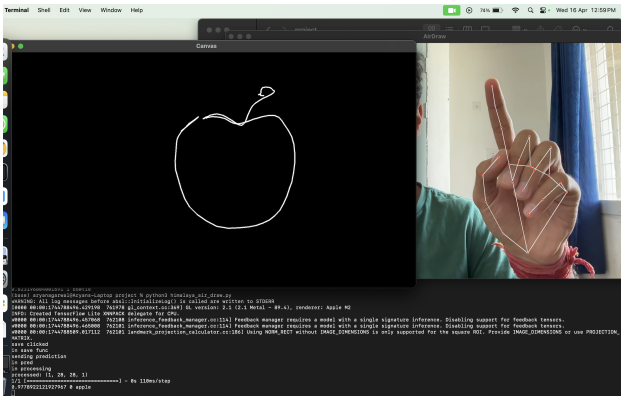

Fig. 24. Air Drawing UI predicting Apple for the object Drawn

3) On technical end following challenges included smooth stroke tracing, managing frame drops , cleaning noisy motion data, inference speed, and ensuring the final data is preprocessed into right dimensions without much distortion (standardized, centered 28 x28 image) for model prediction , and all this in efficient time to have good inference.

VII. Conclusion and Future Work

To optimize inference speed lightweight models are decided to be used. We plan to scale our hybrid model for further improvement with better GPU resources for training and deploy on edge devices. Our system achieves sub-second inference times (50-200 ms per input), making it suitable for edge devices, smart home integrations, educational and design tools. We had a exhaustive understanding of how different models are trained and corresponding challenges and bottlenecks in training and inference. Overall Our UI bridges deep learning and real world interactions using our DoodleVison in an efficient and engaging way. Future work can be explored with better hardware resources to extend model training on transformers to see its emergent properties come into action.

References

[1] A. Wahab, M. Ashraf, and F. Nadeem, "Design and implementation of real-time object detection system based on single-shoot detector and opencv," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 567–574, 2021.
[2] Y. Zhao and S. Wang, "Real-time object detection (yolov5)," 2021, preprint.
[3] J. Zhong, H. Qian *et al.*, "Improved real-time object detection method based on yolov8: a refined approach," 2023, preprint.
[4] J. Lee, B. Varghese, and H. Vandierendonck, "Roma: Run-time object detection to maximize real-time accuracy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 789–802, 2021.
[5] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang, "Deep learning for free-hand sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 10, pp. 2349–2362, 2018.
[6] H. Faseeh, A. Mehmood, M. Shabbir, and F. K. Hussain, "Deep learning assisted real-time object recognition and depth estimation for enhancing emergency response in adaptive environment," *Multimedia Tools and Applications*, 2024.
[7] A. S. Parihar, G. Jain, S. Chopra, and S. Chopra, "Sketchformer: Transformer-based approach for sketch recognition using vector images," *Multimedia Tools and Applications*, vol. 80, pp. 9075–9091, 2021.

[8] V. B. R. Buchireddy *et al.*, "Hand-drawn doodle detection using machine learning," 2022, conference/workshop paper.
[9] G. K. N., A. R. Belagali, M. Rashmi, and R. M. R. Guddeti, "Interactive system for toddlers using doodle recognition," 2022, local conference/workshop paper.
[10] M. Shafiq and Z. Gu, "Deep residual learning for image recognition: A survey," *Artificial Intelligence Review*, 2022.
[11] P. Sethi, S. Duhan, S. Gupta, and G. Jain, "Meta analysis of deep learning models for doodle recognition," 2022, technical Report.
[12] H. Jain, K. Harisinghani, S. Gangar, and M. Kambli, "Doodsearch - opencv with image recognition," 2021, student project or workshop paper.