RVL-CDIP: An Ensemble of Pre-trained CNNs & Transformers for Multiclass Image Classification

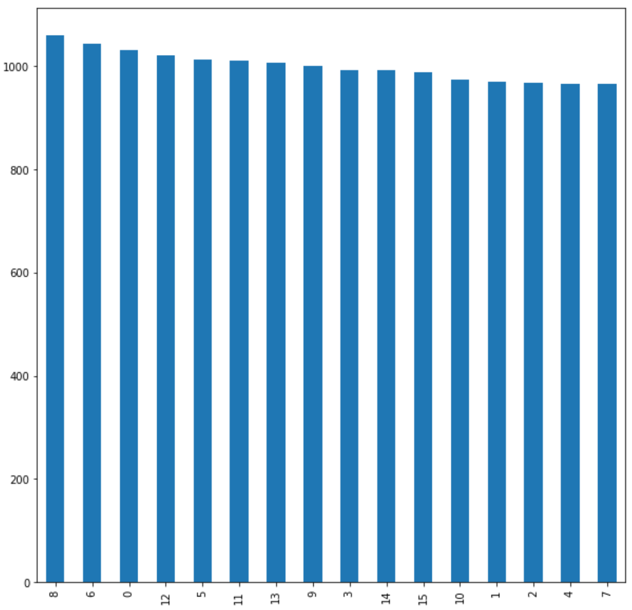
Archit Mangrulkar[1] **Ashish Rekhani**[1] **Raj Shekhar Vaghela**[1] **Devendra Palod**[1] **Deepiha S**[1]

Indian Institute of Technology, Kharagpur.

Abstract

We have a set of grayscale document images and the task is to classify each image into one of the 16 classes or document types. The training dataset which we have used is an RVL-CDIP dataset that consists of 16000 images with ~1000 images belonging to each class. The validation dataset consists of 900 images.

Data Preprocessing



[FIG] The image shows the distribution of images across various classes

Dealing with TIF Images: Renaming the tif files into jpeg files using the rename() under the os module of stdlib of Python 3 because currently, the tensor flow doesn't support .tif files.

Resizing the images: Converting the image files into RGB format using the convert() under the Python Imaging Library because our model takes input only of size 224*224*3 i.e. The input should have three color channels.

Converting into tensors: Transforming the images into tensors by first converting them into a NumPy array and then into tensors using the constant() under TensorFlow so that we can feed them to our model.

Creating Batches: Creating data batches of size 32 using the batch() of the TensorFlow library is done because loading the images in batches requires way less memory as compared to loading the complete dataset. Also if we load the entire dataset then all of its error values are also stored which can decrease the speed of training of our model. Lastly, Mini Gradient Descent is faster than both Batch or Stochastic gradient descent.

Literature Review

This paper[1] introduces us to DenseNet architecture where the layers are inter-connected in a feed-forward fashion. In a classical CNN with number of layers n, we have only one connection between each layer hence a total of n connections. Whereas in Densenet architecture each layer is connected to its following deeper layers hence a total of n(n+1)/2 layers. This paper shows the comparison of experimental results by different models like ResNet, Deeply supervised Net along with DenseNet on datasets like Street View House Numbers (SVHN) dataset and ImageNet.

DenseNet implementation requires a significant amount of GPU memory. This paper[2] introduces us to the method of having pre-allocated Shared Memory Storage locations to avoid quadratic memory growth and reduce the amount of memory consumed by DenseNets during training.

This paper[3] proposes MobileNet as an efficient neural network architecture with depth wise separable convolutions as a replacement for classical convolution layers to meet the requirements for mobile and embedded vision applications. It shows how we can build smaller, faster, and more efficient MobileNets using two hyper parameters called width multiplier and resolution multiplier.

This paper[4] proposes the architecture of MobileNetV2, an extension of MobileNet with linear bottleneck and inverted residual structure. It contains a full convolution layer which has 32 filters associated with it. This is followed by 19 residual bottleneck layers and shows the results of this model on datasets like ImageNet and COCO dataset and compares the results between MbileNetV2 and other models.

In this paper[5], Large MobileNetV3 and Small MobileNetV3 models are introduced for mobile classification, detection and segmentation problems. Proposes different changes that can be made by adjusting the features of MobileNet2 and MnasNet to get a more efficient model.

This paper[6] shows how EfficientNets are more accurate and efficient than ConvNets. The model discussed in the paper, EfficientNet-B7, has an 84.3% top-1 accuracy on the ImageNet dataset even though it is 8.4x smaller and 6.1x faster on inference than the current best ConvNet.

This paper[7] investigates the EfficientNet-B0 model. The network's basic structure is the mobile inverted bottleneck convolution module inherited from MobilenetV2 architecture. This paper discusses the different models of EfficientNet-B0. This is achieved by removing or repositioning modules. This is followed by training all the different models on the ImageNet dataset to demonstrate how the performance of EfficientNet-B0 is affected by SE modules.

This paper[8] discusses the prototypical networks approach few-shot and zero-shot learning. It shows how this approach is simple and more efficient than meta-learning approaches and produces accurate results even without complicated extensions developed for matching networks. It further shows how to generalize prototypical networks to the zero-shot setting, and achieve results on the CUB-200 dataset.

In this paper[9], the authors have developed a unified approach that works for both classical and few-shot learning. In the final classification layer of a trained neural network, the activation vector and the parameter vector have similar structures. And learning a category-agnostic mapping from activations to parameters has helped us in getting better results. The experimental results on ImageNet and MiniImageNet datasets demonstrate the effectiveness of the proposed method for learning category-agnostic mapping.

The paper[10] talks about residual networks and residual learning and connects deep learning with residual networks. It describes the architecture of residual networks and mapping by shortcuts. It discusses the implementation level details and various experiments which are performed related to residual networks.

This paper[11] discusses the detection of breast cancer using a combination of the strength of both the residual networks and inception networks. It talks about both image based and patch based classification for detecting breast cancer and elaborates on some related experiments.

This paper[12] proposes a two-dimensional version of the Sequencer module, where an LSTM is decomposed into vertical and horizontal LSTMs to enhance performance. It talks about why LSTMs are better than vision transformers and also proposes an architecture using LSTMs which is used for some special type of pattern detection.

The paper[13] talks about the supremacy and why LSTMs are better over convolutional neural networks for spectral classification. It also discusses some of the works and experiments which were done in the past to solve the issues related to image classification.

This paper[14] describes how to learn multi-scale feature representations in transformer models for image classification. It proposes a dual-branch transformer to combine image patches (i.e., tokens in a transformer) of different sizes to produce stronger image features.

This paper[15] proposes a technique for producing 'visual explanations' for decisions from a large class of Convolutional Neural Network (CNN)-based models, making them more transparent and explainable.

Experimental Settings

All the experiments were run on Google Colab using a standard K80 GPU and 12 GB of RAM. The libraries used in the various models have been specified in the Requirements.pdf

Method Description & Model Architecture

We have employed pretrained CNN and transformer architectures for the current image classification task. Since starting from scratch can be time-consuming and expensive, we take what a machine-learning model has learned in one domain and apply it to another. This is known as **Transfer Learning**.

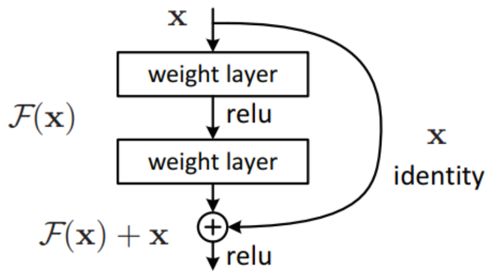
ResNet

Deeper neural networks are more difficult to train, so we use a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We also explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Results show that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.

The Residual Blocks idea was created by this design to address the issue of the vanishing/exploding gradient. We apply a method known as skip connections in this network. The skip connection bypasses some levels in between to link layer activations to subsequent layers. This creates a leftover block. These leftover blocks are stacked to create resnets.

The strategy behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping.

F(x) := H(x) - x which gives H(x) := F(x) + x.



Dense Net

DenseNet is a convolutional neural network where all the layers are connected to all the layers that are deeper in the network. This is done to improve the accuracy of a model by handling the problem of vanishing gradients by ensuring maximum information is shared between all the layers. Because of these connections across different layers, we call this the Dense Convolutional Network (DenseNet). Each layer in the DenseNet model takes outputs from all the previous layers in the network as its input.

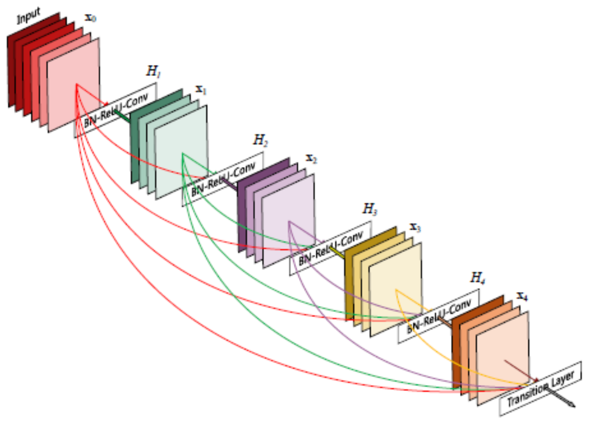
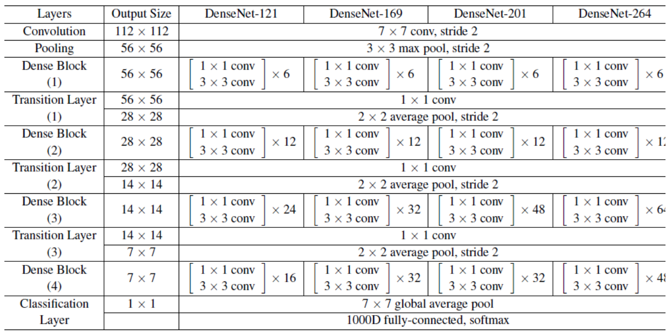


FIG: A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature maps as input.

Apart from the primary convolutional layers, it has two more important blocks known as the Dense Block and Transition layers.

- **Convolutional Block:** Each convolutional block processes the input in the following order: BatchNormalization, which is followed by ReLU activation and finally passed through the actual Conv2D layer.

- **Dense blocks:** Each dense block consists of convolution layers. Dense blocks and Transition layers are placed alternatively. Every layer in a dense block is connected to all of its following layers and receives the feature maps of all the layers before it. The final dense block is followed by a classification layer.
- **Transitional Block:** The layers between the dense blocks which do the convolution and pooling are called transition layers. DenseNet has a batch-norm layer, 1x1 convolution and a 2x2 average pooling layer as its transition layers.



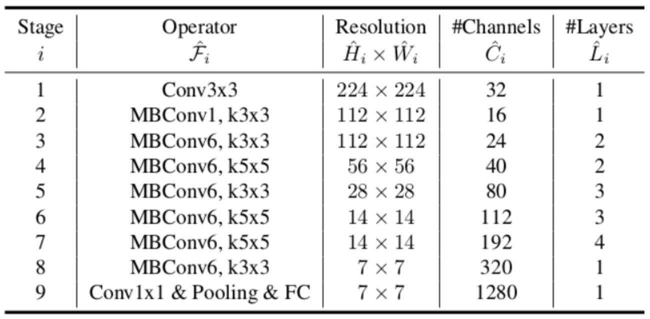
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 1$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 4$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

FIG: DenseNet architecture for ImageNet

Efficient Net

EfficientNet is a convolutional neural network(CNN) architecture which uses a scaling method with a compound coefficient to uniformly scale the network depth, width and resolution dimensions. This increases the accuracy and efficiency as measured on the floating-point operations per second (FLOPS) basis.

Similar to the MobileNet V2 architecture, EfficientNet also uses mobile inverted bottleneck convolution (MBConv).



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

FIG: EfficientNet-B0 baseline network

The EfficientNet scaling method uniformly scales different dimensions of the network with a set of fixed scaling coefficients because if the input image is large, then the network also needs more layers and more channels to capture all the features of the bigger image precisely.

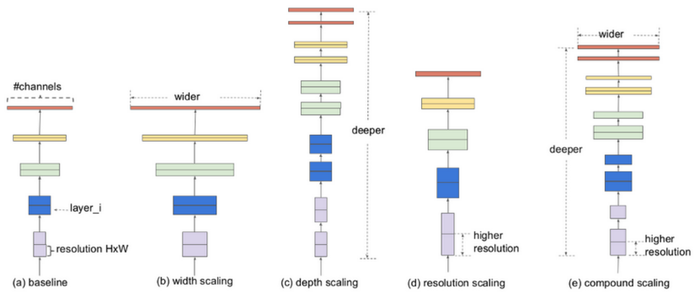
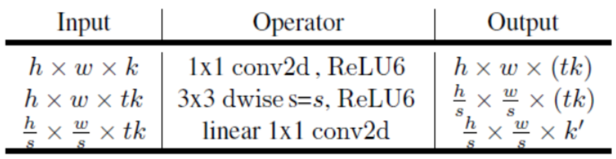


FIG: **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Mobile Net

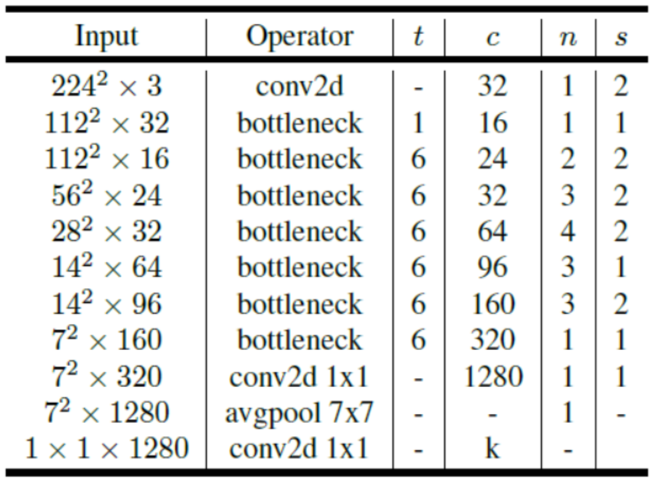
The objective of the MobileNet model is to make neural networks much lighter and portable for mobile and embedded vision applications and it is primarily done by using depth-wise separable convolutions which significantly reduces the number of parameters. MobileNets are small, less power-consuming models made to meet the resource constraints of various use cases.

Apart from the older version MobileNet v2 has two new additions, inverted residuals with a linear bottleneck. MobileNet-v2 has 53 layers. It has two types of blocks, the residual block and another block for downsizing and each of these blocks has three layers. the first layer is a 1x1 convolution layer with ReLU6, the second layer is a 3x3 depthwise convolution layer with ReLU6 and the final third layer is a linear 1×1 convolution layer.



Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

FIG: Bottleneck residual block transforming from k to k0 channels, with stride s, and expansion factor t.

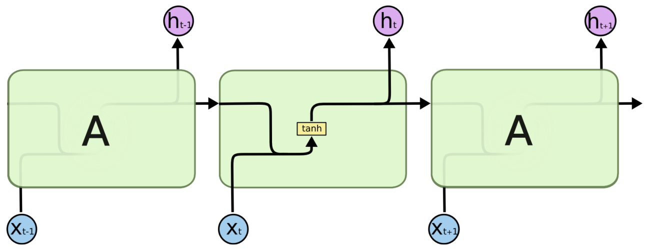


Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

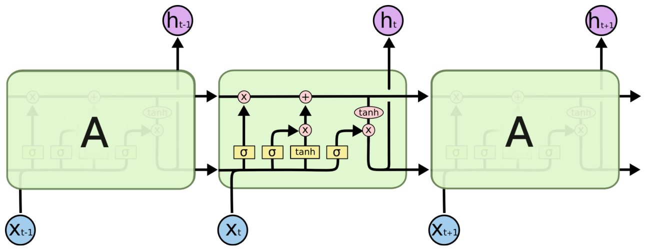
FIG: MobileNetV2 Overall Architecture

LSTM
LSTMs are created to prevent the long-term reliance issue. They don't struggle to learn; rather, remembering information for extended periods of time is basically their default behavior.

All recurrent neural networks have the shape of a series of neural network modules that repeat. This recurring module in typical RNNs will be made up of just one tanh layer, for example.



Although the repeating module of LSTMs also has a chain-like structure, it is structured differently. There are four neural network layers instead of just one, and they interact in a very unique way.



1. CNN with attention
   A Convolutional Neural Network(CNN) is a kind of neural network architecture which is specifically used for image recognition, object classification and pattern recognition which involves the processing of image-based pixel data.

2. Ensemble models
   We create multiple models and combine them to produce results instead of a single model to make our results more accurate. The different types of ensemble models are bagging, boosting and stacking.

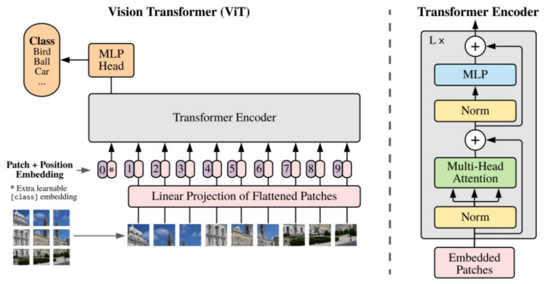
   - In bagging, we take the average of all the results from different models as the final output.
   - In boosting, we change the training dataset to focus more on the examples that the previous models have gotten wrong i.e its main property is to correct prediction errors.
   - In stacking, we divide our training dataset into various units and are fed to different base models and the results from these are then combined into one by a meta or first-level model.

Transformers - Vision Transform
As shown by language models like BERT and GPT-3, transformers first found use in natural language processing (NLP) activities. The usual image processing system, in contrast, makes use of a convolutional neural network (CNN). Xception, ResNet, EfficientNet, DenseNet, and Inception are a few well-known projects.

Transformers measure what is known as attention—the connections between pairs of input tokens (words in the case of text strings). The price is a quadratic function of the token count. The pixel serves as the fundamental analytical unit for images. However, the amount of memory and processing required to compute relationships for each pair of pixels in a typical image is prohibitive. Instead, at a far lower cost, ViT computes relationships between pixels in various small portions of the image (such as 16x16 pixels). The portions are organized into a sequence using positional embeddings. The vector embeddings can be learned. Each portion is multiplied by the embedding matrix and put in a linear order. The transformer receives the outcome together with the position embedding.

The class token plays a crucial role in classification tasks, same like in BERT. A unique token that, after being affected by all the others, serves as the final MLP Head's sole input.

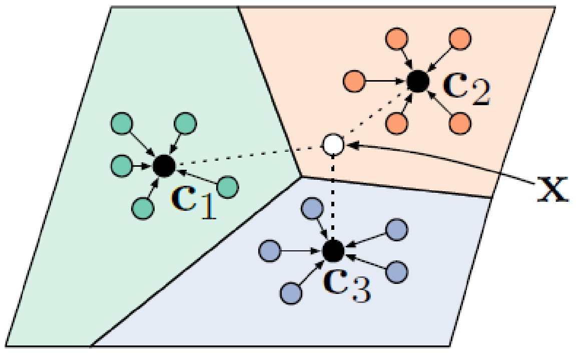
The most popular architecture for picture classification transforms the different input tokens only using the Transformer Encoder. The decoder component of the conventional Transformer Architecture is employed in more applications, though.



Few Shot

Few-shot image classification is a way of doing image classification with only a few examples for each classification class(typically < 6 examples). The few-shot learning technique is extremely useful when you want machines to learn rare cases. Few-shot learning uses the N-way-K-shot classification method to differentiate between N classes given there are K examples in the support set given for each class, where the task is to classify the query or test images into the N classes. We can use few-shot learning for not only Image Classification but also for Object Detection, Action Recognition, Semantic Segmentation, Natural Language Processing, Image Generation, and many more tasks.

Prototypical Networks classify new classes (not part of training classes) based on their similarity to a small number of examples per class. This is one of the simplest algorithms to solve FSL. Let's discuss the example of a 3-way 5-shot classification. Here, We project the support images to the feature space just like this plot shown below and which can be done using a feature network like the ResNet network. Then the few-shot prototypes c1, c2 and c3 are computed as the mean of embedded support images for each class i.e we take the average of feature vectors generated by the network to get c1, c2 and c3. Then the new query image is projected on the feature or embedding space and compared to these prototypes using Euclidean distance to classify as one of these classes.



Some of the other famous Few-Shot Image Classification algorithms are Model-Agnostic Meta-Learning (MAML), Matching Networks and Relation Networks.

Few-Shot Learning is also an example of meta-learning, where the model is trained using a series of training tasks in the meta-training phase, and then it is able to classify known to unknown but related tasks with help of only a few examples in the meta-testing phase. In meta-learning the training set is called the support set and the test set is called the query set. Few-shot training is different from the traditional methods of training machine learning models in terms of training data used, as traditional training methods use a large amount of training data.

Experimental Results

Here we compare the training and test accuracy of the various models deployed.

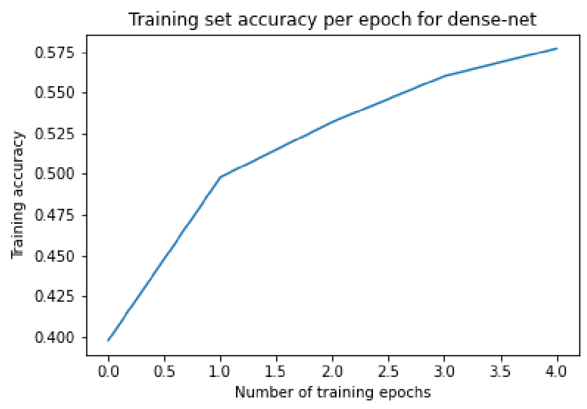


FIG: Training accuracy for DenseNet model plotted against epochs. Here at the end of 5 epochs we achieve an accuracy of 57.7%.

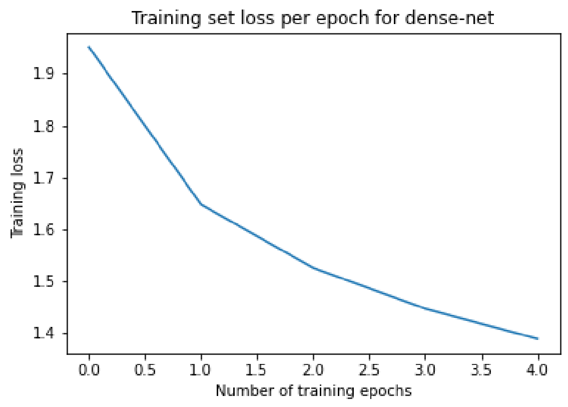


FIG: Training loss for DenseNet model plotted against epochs. Here we notice a steep decline in the loss for first epochs from 1.95 till 1.62 but the curve gets less steep as the epochs increase.

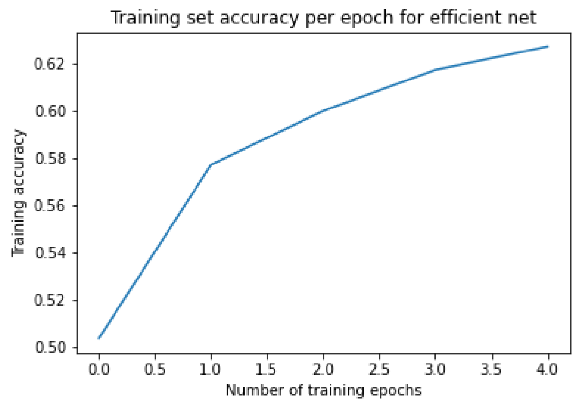


FIG: Training accuracy for EfficientNet model plotted against epochs. Here at the end of 5 epochs we achieve an accuracy of 66.6%.

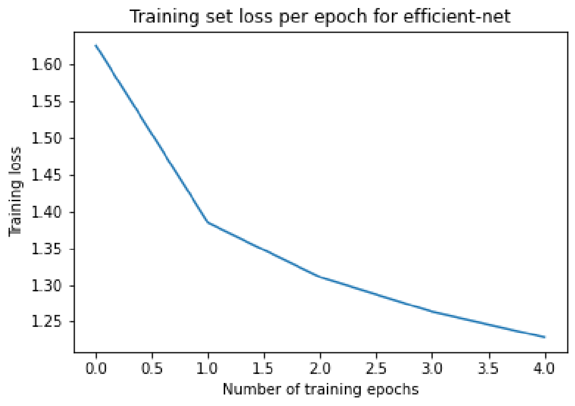


FIG: Training set loss for Efficient net model plotted against epochs.

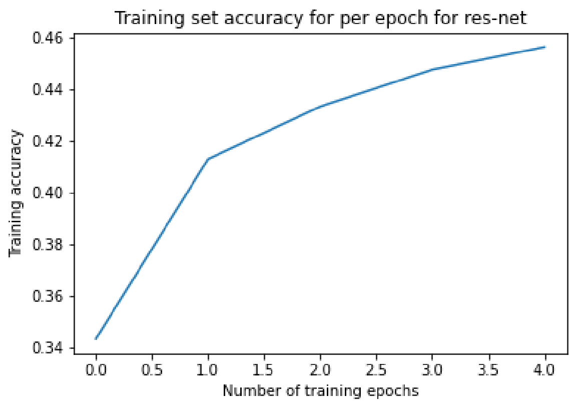


FIG: Training accuracy for ResNet model plotted against epochs. Here at the end of 5 epochs we achieve an accuracy of 45.63%.

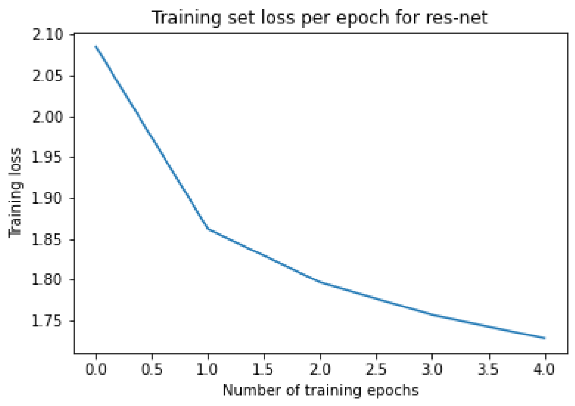


FIG: Training loss for ResNet model plotted against epochs. Here we notice a steep decline in the loss for first epochs from 2.08 till 1.85 but the curve gets less steep as the epochs increase.

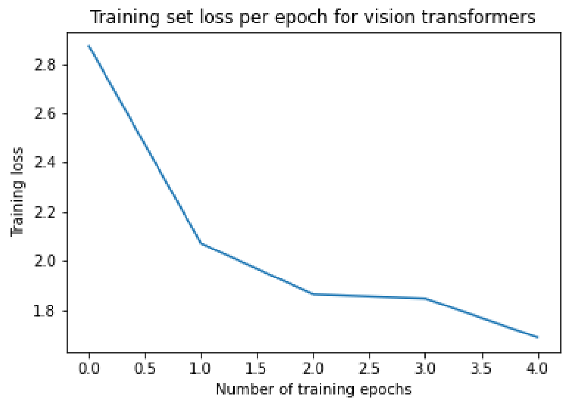


FIG: Training loss for vision-transformer model plotted against epochs. Here we notice a steep decline in the loss for first epochs from 2.08 till 1.85 but the curve gets less steep as the epochs increase.

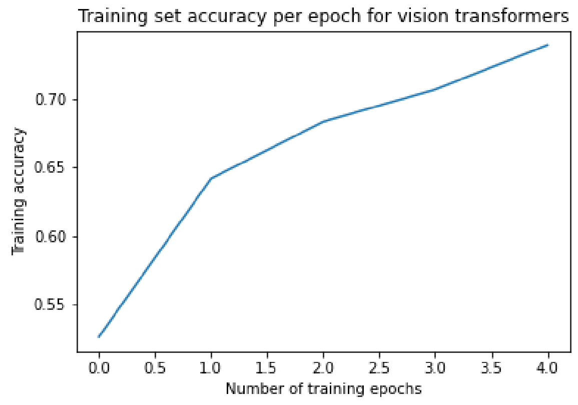


FIG: Training accuracy for vision-transformer model plotted against epochs. Here at the end of 5 epochs we achieve an accuracy of 73.87%.

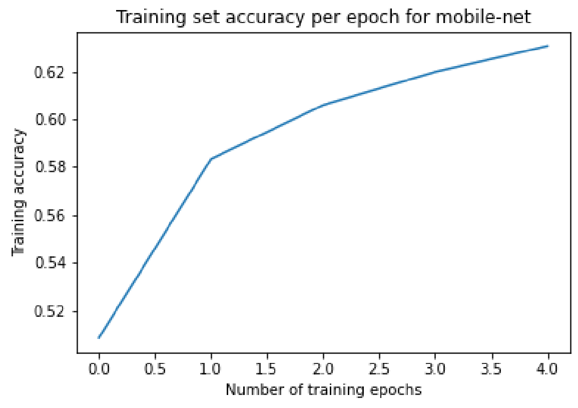


FIG: Training accuracy for MobileNet model plotted against epochs. Here at the end of 5 epochs we achieve an accuracy of 63.06%.

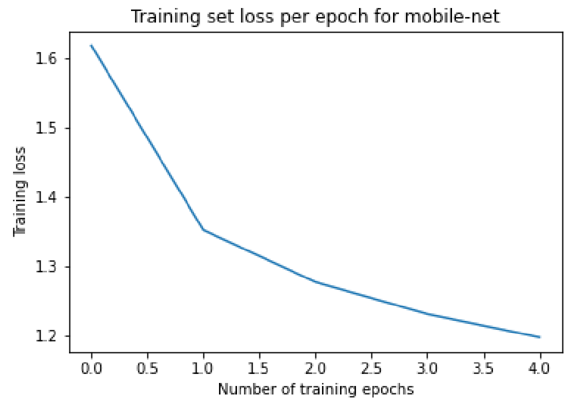


FIG:Training loss for Mobile Net model plotted against epochs. Here we notice a steep decline in the loss for first epochs from 1.6 till 1.35 but the curve gets less steep as the epochs increase.

*FewShot could only be trained on very small batches and thus the high accuracy. Gradients couldnt be computed as they became NaN after a couple iterations of the Reptile algorithm.
**LSTM would give loss NaN after few epochs so accuracy could not be improved

Model	Training Accuracy	Test Accuracy
ResNet	0.4563	0.486
MobileNet(up to 4000 images)	0.6306	0.631
MobileNet(entire set)	**0.6306**	**0.720**
DenseNet	0.5773	0.240
Efficient Net	0.6274	0.606
Few Shot*	0.937	-
Vision Transformer	**0.7387**	**0.752**
Ensemble by Loss	**-**	**0.770**
LSTM**	0.06	-
Ensemble by Accuracy	-	0.669

Novelty and Innovations

Ensemble Model: We have created an ensemble model combining the various CNN and Transformer models by taking a weighted average of the logits according to their losses and accuracies of these models. The weights have been assigned manually but these weights could also.

Ensemble by Accuracy: We choose the 3 models with highest accuracy (Efficient Net, Mobile Net and Vision transformer) and assign them weights in proportion of their testing accuracy. Since these are the best performing models, the resultant model is expected to perform better and counteract the inaccuracies of each other.

Ensemble by Loss: We choose the 3 models with highest accuracy (Efficient Net, Mobile Net and Vision transformer) and assign them weights in inverse proportion of their testing loss. Since these are the best performing models, the resultant model is expected to perform better and counteract the inaccuracies of each other.

RestNet with supervised contrastive learning: We have used ResNet with Supervised Contrastive Learning. It is an alternative loss function to cross entropy for multi class image classification.Clusters of points belonging to the same class are pulled together in embedding space, while simultaneously pushing apart clusters of samples from different classes.

LSTM: Inspired by the literature review of LSTM on image classification, we tried implementing a LSTM model for multi class image classification. The model gives NaN loss and training accuracy of 6.08% on epoch 1 and would not improve on further epochs. This happened mainly because of low temporal correlation between the images.

FewShot: We have implemented the Reptile Algorithm to perform Model Agnostic Meta Learning. In each iteration of the algorithm, we take a mini batch of 25 examples and using a ConvNet we make predictions, compute gradients and update the model weights and biases. We repeat this whole process for 2000 meta iterations. However, after a few iterations it suffers a problem of vanishing gradients.

Future Works and Improvements

GPU Constraints: Due to limited availability of GPU time (as restricted by Collab) the models could only be trained up to 5 epochs. With a better GPU, the training could have been more extensive and faster as well providing better results.

Ensemble Model: We have created an ensemble model combining the various CNN and Transformer models by taking a weighted average of the logits according to their losses and accuracies of these models. The weights have been assigned manually but these weights could also have been obtained by Grid Search (Hyperparameter Tuning where we check for each of the possible combinations of weights randomly and select the one which gives us the best performance) to further improve the robustness and the performance of our model. Another approach could be defining a new ensemble model that uses a set of activation functions- RelU, tanH & sigmoid and instead of directly performing a weighted average of logits, the model assigns a different activation function for each of the model logits and then combines then linearly according to weights obtained after the Grid Search.

Knowledge Distillation: In this approach, we first train a deep neural network on the given data (ground labels). The subsequent models are then trained on the ground labels as well as the data from the previous models. The loss function of the subsequent models are not only a function of ground labels but also that of the new model with respect to the previous models.

Conclusion

We have used multiple models mentioned in the report to carry out the document classification of the

RVL-CDIP dataset. We have compared the performance of each of the models and have also combined the results of various models to create an ensemble model. Vision Transformer performed the best in terms of test accuracy of 75.2%. We have also created an Ensemble model which has a test accuracy of

References

[1]Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q., "Densely Connected Convolutional Networks", <i>arXiv e-prints</i>, 2016.

[2]Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., and Weinberger, K. Q., "Memory-Efficient Implementation of DenseNets", <i>arXiv e-prints</i>, 2017.

[3]Howard, A. G., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", <i>arXiv e-prints</i>, 2017.

[4]Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C., "MobileNetV2: Inverted Residuals and Linear Bottlenecks", <i>arXiv e-prints</i>, 2018.

[5]Howard, A., "Searching for MobileNetV3", <i>arXiv e-prints</i>, 2019.

[6]Tan, M. and Le, Q. V., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", <i>arXiv e-prints</i>, 2019.

[7]V.-T. Hoang, K.-H. Jo, Practical analysis on architecture of EfficientNet, in: 2021 14th International Conference on Human System Interaction, HSI, 2021

[8]Snell, J., Swersky, K., and Zemel, R. S., "Prototypical Networks for Few-shot Learning", <i>arXiv e-prints</i>, 2017.

[9]Qiao, S., Liu, C., Shen, W., and Yuille, A., "Few-Shot Image Recognition by Predicting Parameters from Activations", <i>arXiv e-prints</i>, 2017.

[10]He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition", <i>arXiv e-prints</i>, 2015.

[11]Zahangir Alom, M., Yakopcic, C., Taha, T. M., and Asari, V. K., "Breast Cancer Classification from Histopathological Images with Inception Recurrent Residual Convolutional Neural Network", <i>arXiv e-prints</i>, 2018.

[12]Tatsunami, Y. and Taki, M., "Sequencer: Deep LSTM for Image Classification", <i>arXiv e-prints</i>, 2022.

[13]Liu, Q., Zhou, F., Hang, R., and Yuan, X., "Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification", <i>arXiv e-prints</i>, 2017.

[14]Chen, C.-F., Fan, Q., and Panda, R., "CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification", <i>arXiv e-prints</i>, 2021.

[15]Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", <i>arXiv e-prints</i>, 2016.

[16]Luo, Y., Zou, J., Yao, C., Li, T., and Bai, G., "HSI-CNN: A Novel Convolution Neural Network for Hyperspectral Image", <i>arXiv e-prints</i>, 2018.

Contributions:-

1. Archit Mangrulkar (20CS10086):
    a. Model Building (Vision Transformer, EfficientNet, Few Shot Learning)
    b. Model Training (Vision Transformer, EfficientNet, Few Shot Learning)
    c. Model Testing (Vision Transformer, EfficientNet, Few Shot Learning)
    d. Model Evaluation (Vision Transformer, EfficientNet, Few Shot Learning)
    e. Literature Review on Vision Transformers & Few Shot Learning
    f. Ensemble by Loss & Ensemble by Accuracy of best performing models
    g. Report

2. Ashish Rekhani (20CS10010):
    a. Model Building (Vision Transformer, EfficientNet, LSTM)
    b. Model Training (Vision Transformer, EfficientNet, LSTM)
    c. Model Testing (Vision Transformer, EfficientNet, LSTM)
    d. Model Evaluation (Vision Transformer, EfficientNet, LSTM)
    e. Literature Review on LSTM, Efficient Net
    f. Report

3. Raj Shekhar Vaghela(22CS60R32):
    a. Dataset Preprocessing
    b. Model Building (MobileNet)
    c. Model Training (MobileNet)
    d. Model Testing (MobileNet)
    e. Model Evaluation (MobileNet)
    f. Literature Review on MobileNet

4. Devendra Palod (20CS10024):
    a. Dataset Preprocessing
    b. Literature Review on DenseNet
    c. Model Building (DenseNet)
    d. Model Training (DenseNet)
    e. Model Testing (DenseNet)
    f. Model Evaluation (DenseNet)
    g. README Documentation

5. Deepiha S (20CS30015):
    a. Model Building (ResNet)
    b. Model Training (ResNet)
    c. Model Testing (ResNet)
    d. Model Evaluation (ResNet)
    e. Literature Review on MobileNet and ResNet
    f. Report
    g. README Documentation