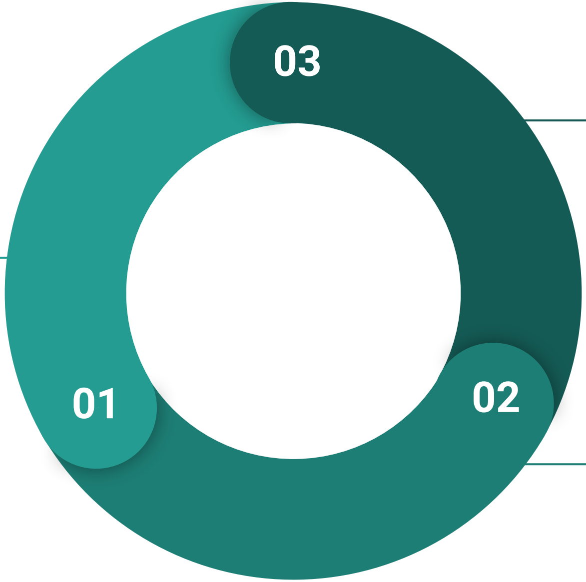
Low Rank Adaptation for Image Super Resolution

Team Members:
Anirudh Muthuswamy (NUID - 002783250)
Gugan Kathiresan (NUID - 002756523)
Aditya Varshney (NUID - 002724777)

Problem Statement



In the realm of image processing, achieving single image super-resolution remains a fundamental challenge.

In recent times, the size and complexity of SOTA Deep Learning based models has been increasing at perplexing rate.

Training such models for domain specific tasks requires an intensive fine-tuning process.

We aim to apply Low Rank Adaptations (LoRA) to these complex models which could result in significant decrease in compute as well as training time requirements.

Related Work

Super Resolution

- SRCNN: First CNN based architecture
- RCNN: Residual Based CNN
- SRGAN: First GAN for Super Resolution
- ESRT: First Transformer based network
- Hybrid Attention Transformer (HAT): leverages channel attention and window-based self-attention schemes to capture global and local context
- Latent Diffusion Models (LDM)

Transfer Learning

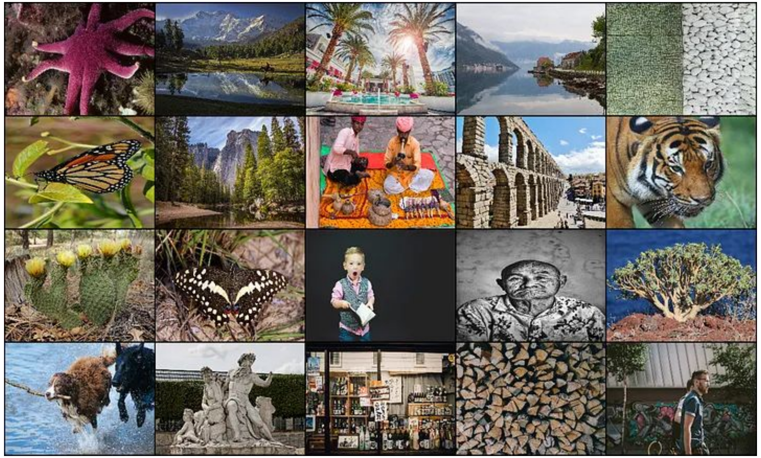
- Fine-tuning : Train the model on a new dataset, initialising the weights from a pre trained task.
- Adapter Layers : Add Adapter Layers which learn domain specific task into an existing model.
- Prefix-training : optimize a small task-specific vector (prefix) and keep the model parameters frozen

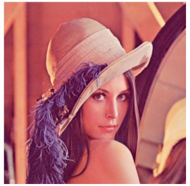
Fine-Tuning

- Huge models require tremendous amount of compute for training.
- Fine-tuning is a long standing method to transfer learn a model
- We use model parameters which have been trained on a particular task and further train the model for a fewer number of epochs but on a different task.
- This leads to the model converging into the newly learnt task

Dataset Used

- We use pre-trained weights trained on a combination of datasets like DIV2K, Flickr2K, and OutdoorSceneTraining.
- For Domain Transfer, we evaluate the model on
    - FloodNet: High Res, Aerial images
    - Set4
    - Set15





Methodology

In our approach, we take the following steps to achieve low complexity SISR and evaluate its efficacy:

1. Setup the ESRGAN with original pretrained weights
2. Create low rank adapters on a different dataset and finetune on the same
3. Conduct an extensive ablation study on the effect low rank adapters have on the model parameters

Latent Diffusion Models

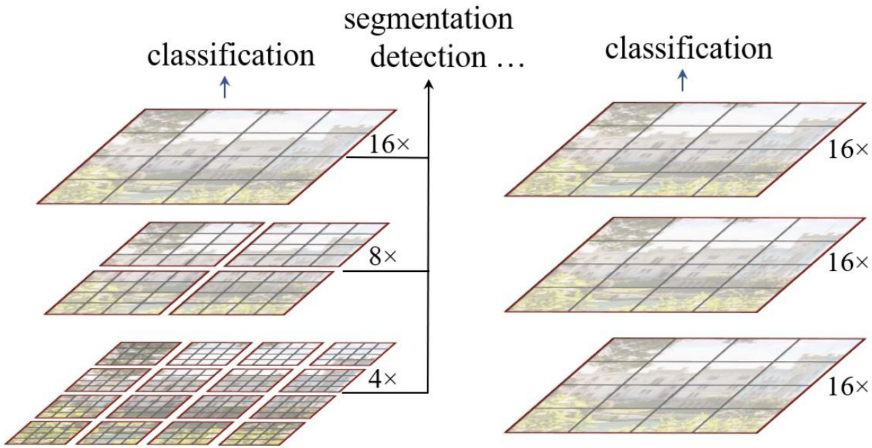
Ground Truth

Output

- Designed to capture the dynamics of underlying processes through the diffusion of latent variables.
- Excel at capturing complex temporal dynamics in underlying processes
- However, lack the ability to generalize well on unseen data. For example, the model hallucinates when a completely different image is passed through it

Swin Transformer

- Builds hierarchical feature maps by merging image patches
- Adds self attention to patches which help capture local context
- Cross-window connection results in global attention is used to capture global context
- Despite the efficiency improvements, attention mechanisms in transformers can still be computationally demanding



(a) Swin Transformer (ours)

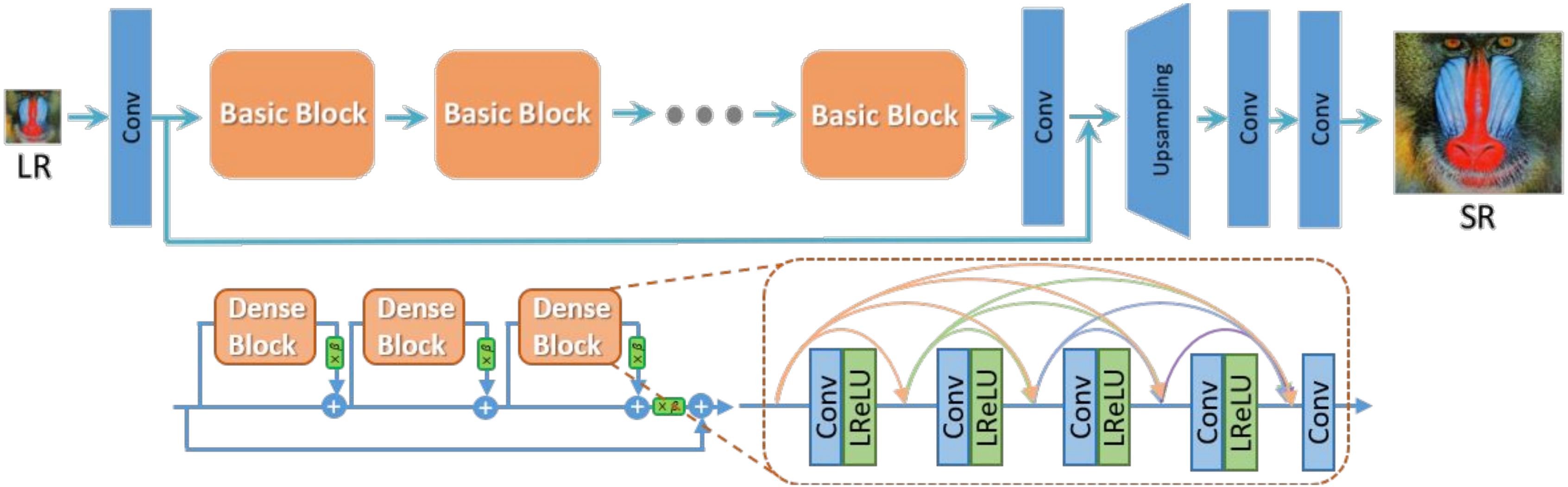
(b) ViT

Enhanced SRGAN (ESRGAN)

An improvement from the SRGAN

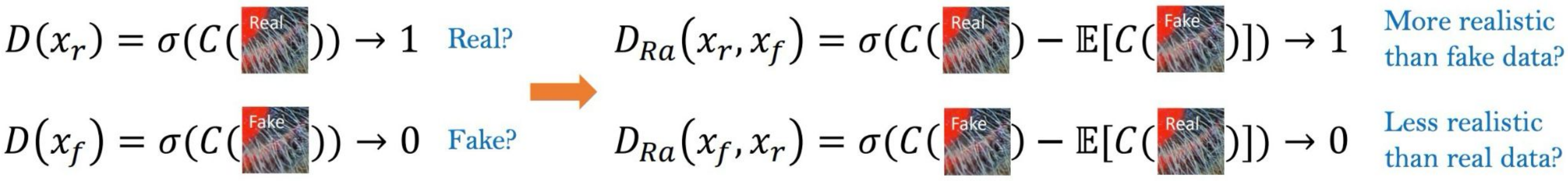
- Residual-in-Residual Dense Block (RRDB)
    - Remove BatchNorms from the Residual Dense Block
- Use of a Relativistic Discriminator
- Adversarial and Perceptual Loss

ESRGAN model architecture



Relativistic Discriminator

- The Standard GAN discriminator classifies if the image received by it is fake or real
- A Relativistic GAN discriminator classifies if the image is more realistic than a fake or less ie. estimates the probability that the given real data is more realistic than a randomly sampled fake data.



a) Standard GAN

b) Relativistic GAN

Generator Loss

- Previously defined on the activation layers of a pre-trained deep network, where the distance between two activated features is minimized
- Instead, we use features before the activation layers
  - the activated features are very sparse, especially after a very deep network which lead to weak supervision
  - using features after activation also causes inconsistent reconstructed brightness compared with the ground-truth image

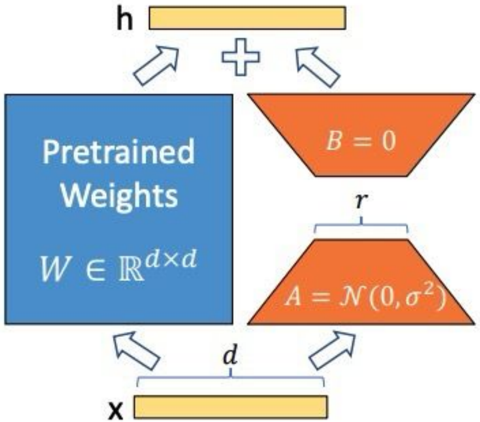
$$L_G = \underbrace{L_{\text{percep}}}_{\text{Perceptual loss}} + \lambda \underbrace{L_G^{Ra}}_{\text{Adversarial Loss}} + \eta \underbrace{L_1}_{\text{Content Loss}},$$

Low Rank Adaptation (LoRA)

- Adapter based method to assist in fine-tuning.
- Weights are essentially huge matrices which have ranks, (number of linearly independent columns)
- If a column is linearly independent, it means that it can't be represented as a combination of other columns in the matrix.
- On the other hand, a dependent column is one that can be represented as a combination of one or more columns in the same matrix.
- You can remove dependent columns from a matrix without losing information.

Low Rank for Fine-tuning

- When fine-tuning an model for a downstream task, you don't need the full-rank weight matrix.
- Hence we can preserve most of the learning capacity of the model while reducing the dimension of weight parameters
- In LoRA, we create 2 weight matrices, one transforms input parameters from original dimension to low -rank dimension, and the other transforms from low rank to output dimension
- Modifications are made to the LoRA parameters, which are now much fewer than the original weights.
- This is why they can be trained much faster and at a fraction of the cost of doing full fine-tuning.
- At inference time, the output of LoRA is added to the pre-trained parameters to calculate the final values.



$$h = W_0 x + \Delta W x = W_0 x + BAx$$

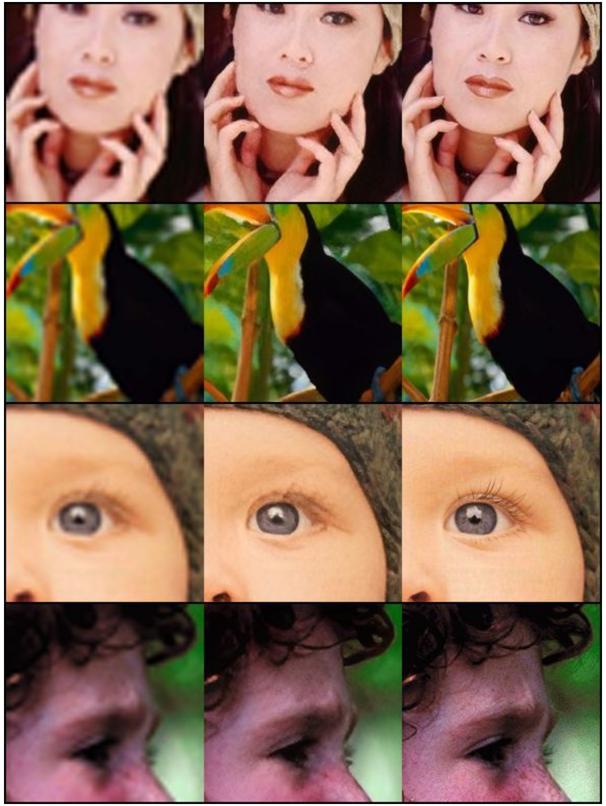
Results

- As LoRA freezes the original weight parameters, there exists a significant drop in number of trainable parameters.
- We observe a 81% decrease in number of trainable parameters, with a significant increase in PSNR values for each independent adapter.
- Furthermore, we also observe a significant decrease in inference time on each dataset.

Experiment	Inference Time ↓
FloodNet	30%
Set5	35%
Set14	27%

Evaluation on Set 5

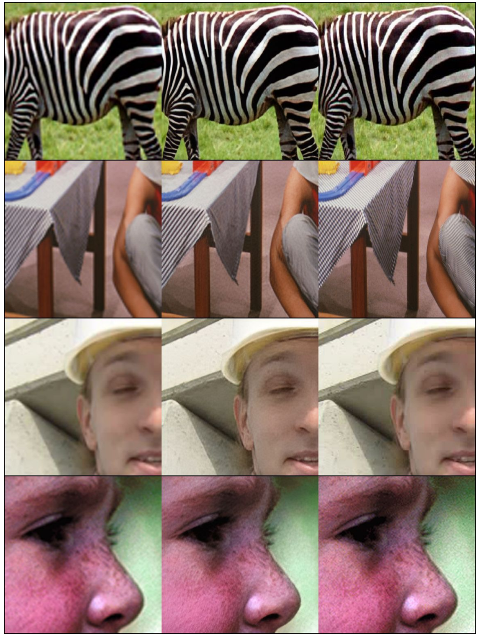
Experiment	PSNR	SSIM	Inference Time(s)
w/o LoRA	23.72	0.684	0.94
w LoRA	26.61	0.775	0.64



Input (LR) Result (SR) GT (HR)

Evaluation on Set 14

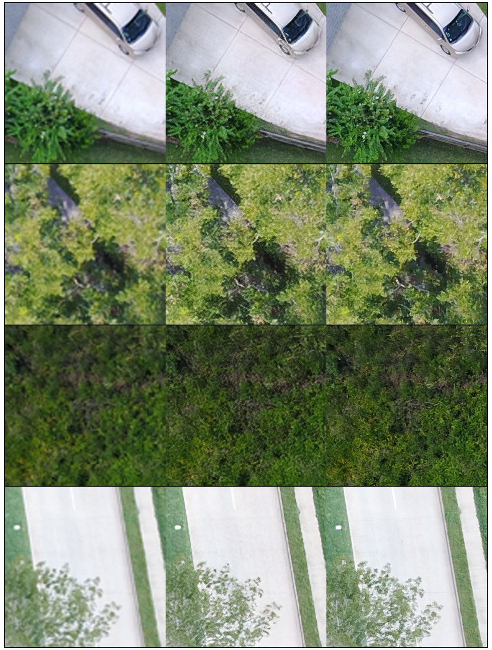
Experiment	PSNR	SSIM	Inference Time(s)
w/o LoRA	24.51	0.711	1.31
w LoRA	28.25	0.798	0.96



Input (LR) Result (SR) GT (HR)

Evaluation on FloodNet

Experiment	PSNR	SSIM	Inference Time(s)
w/o LoRA	24.18	0.536	10.10
w LoRA	25.45	0.601	7.39



Input (LR) Result (SR) GT (HR)

Future Scope

- Work with Transformer / Diffusion based architectures
- Perform experiments on datasets from varied domain

Thank you!