BTP code: B24PBN03

Guide : Dr. Pavan Kumar BN

DECENTRALIZED FEDERATED LEARNING

Decentralized Federated Learning (DFL) for Fog enabled UAV-as-a-Service architecture

Siddhartha V S20210010236

Harshitha K S20210010116

Vishwajith S S20210010197

CONTENTS

INTRODUCTION

Fully Decentralized Learning: The DFL architecture conducts federated learning (FL) over multiple UAVs in a fully decentralized manner, eliminating the need for a central parameter server for global model aggregation.2

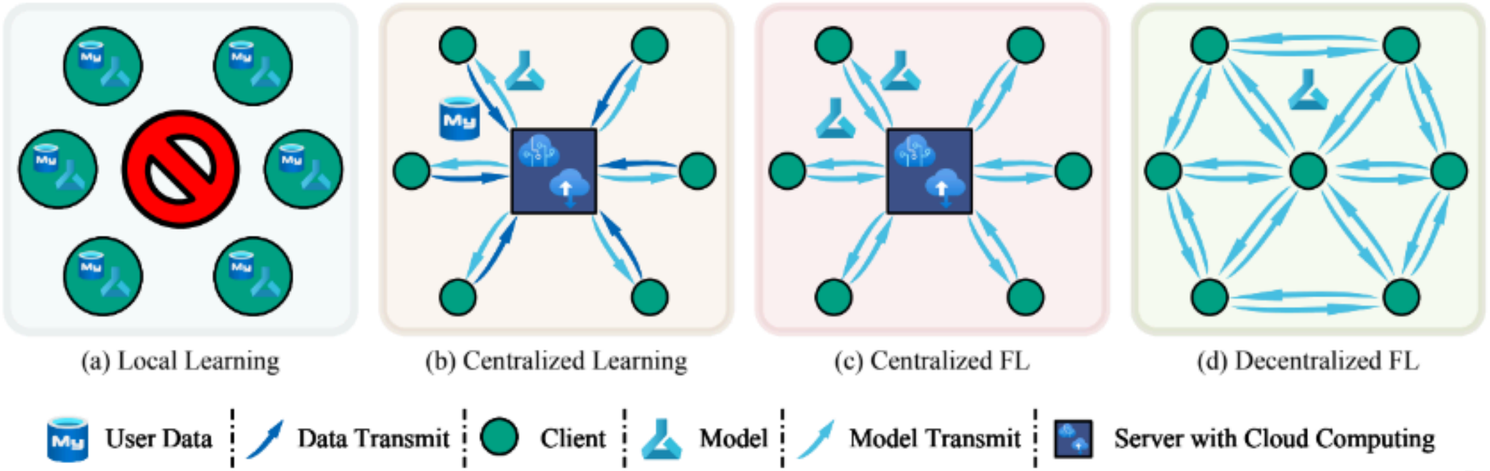
Local Model Updates: Each UAV trains a local FL model using its own dataset and updates its local model parameters based on the received model weights from its neighboring UAVs.

Neighborhood Communication: UAVs exchange their model weights with neighboring UAVs in a one-hop communication fashion. For example, UAV i exchanges model weights with UAVs i + 1, i + 2, and i + 3.

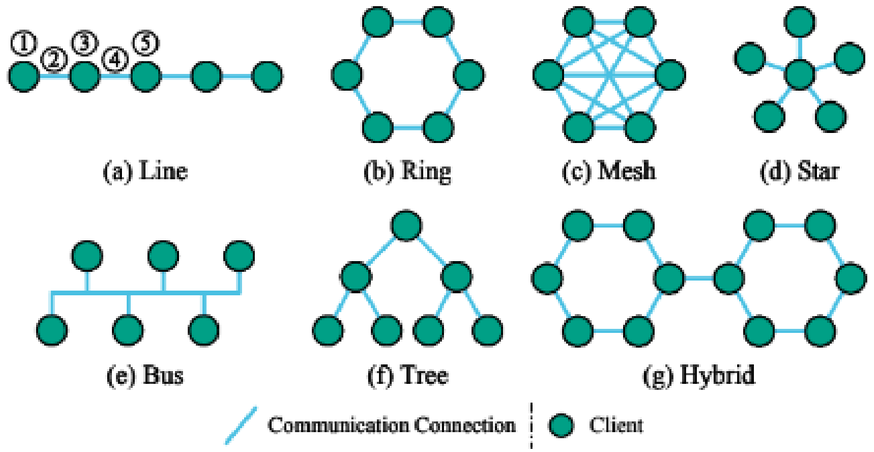
Aggregated Local Model: Upon receiving model weights from neighboring UAVs, each UAV aggregates these weights with its own current model weights to generate an aggregated local model.

Continuous Model Refinement: The process repeats iteratively, with each UAV updating its local model based on the aggregated local model, contributing to a continuous refinement of the FL models across the UAV network [1]

Decentralized Federated Learning



[5]



DATASET DETAILS

- We used publicly available AIDER(Aerial Image Database for Emergency Response applications) dataset and Disaster Images Datasets available on the zenodo platform and kaggle
- It consists images of 5 classes Collapsed Building, Smoke/fire, and,flood,accidents,normal consists of around 8,200 images(all classes combined) after augumentation (before augmenting around 6,500) images

Some Important Points

- Use TCP connection, dont go for UDP connection as the packets may arrive out-of-order and there will be packet loss, use the reliable framework

- Start the servers at a time and their interdependent connections after all the servers are started

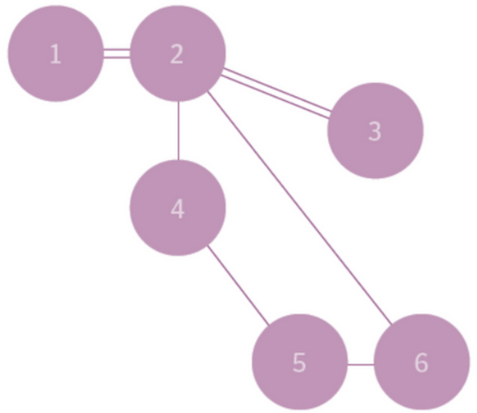
- Dont miss on taking max of the recieved paramteres because nodes may go down and we dont want to over execute transactions on a particular node

Our Setup of servers

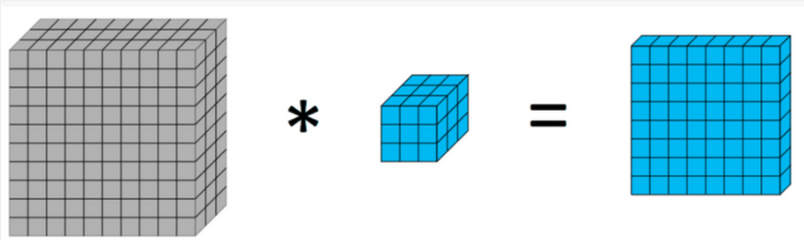
- Our Setup and Configuration : We implemented a Sparse Mesh Topology with 5 Nodes in the network setup and each node is a server which participates in the DFL. and shares it's weights to other "K" random nodes

- Why Sparse Mesh: Our previous Comparision between Mesh and Ring concluded that Mesh yields more accuracy relatively

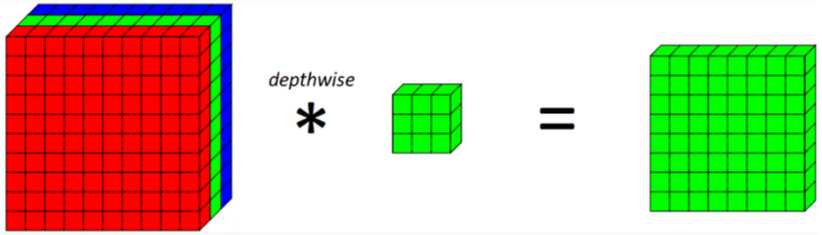
- But it also takes/consumes more power, so to strike good balance between these 2 things, we went with something in the middle (**Sparse Mesh**)



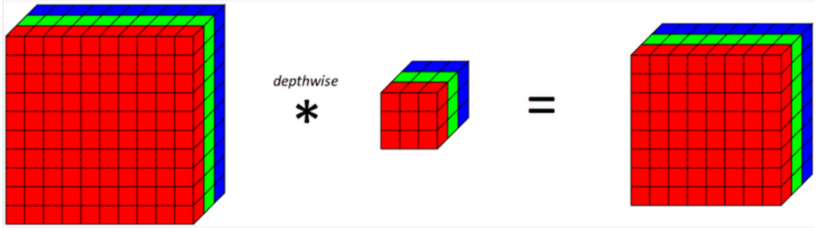
Normal Convolution Vs Depth Wise



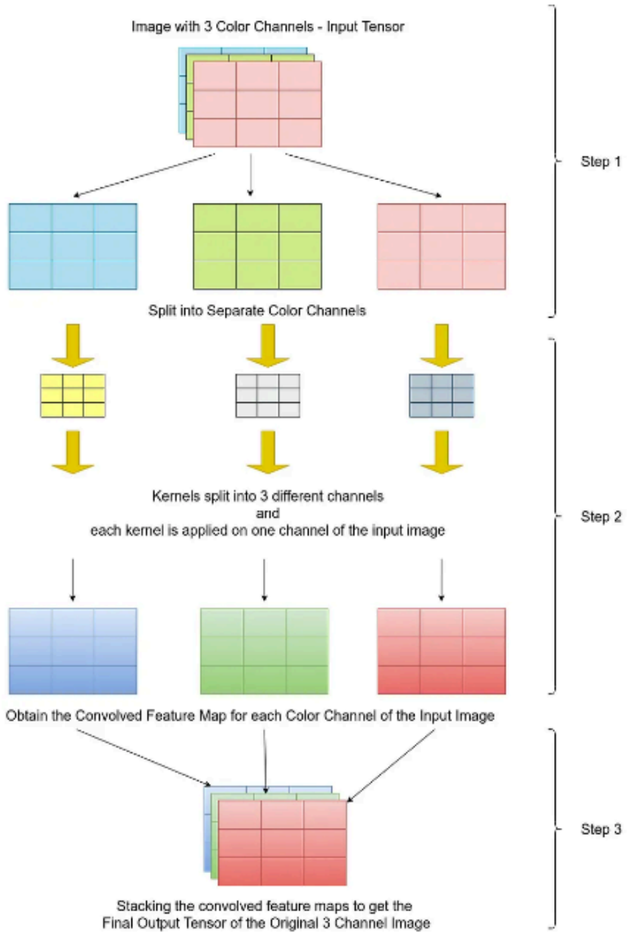
Applying a 3×3 kernel on a 10x10x3 outputs an 8x8x1 volume



Applying a depthwise 3x3 kernel on the green channel in this example

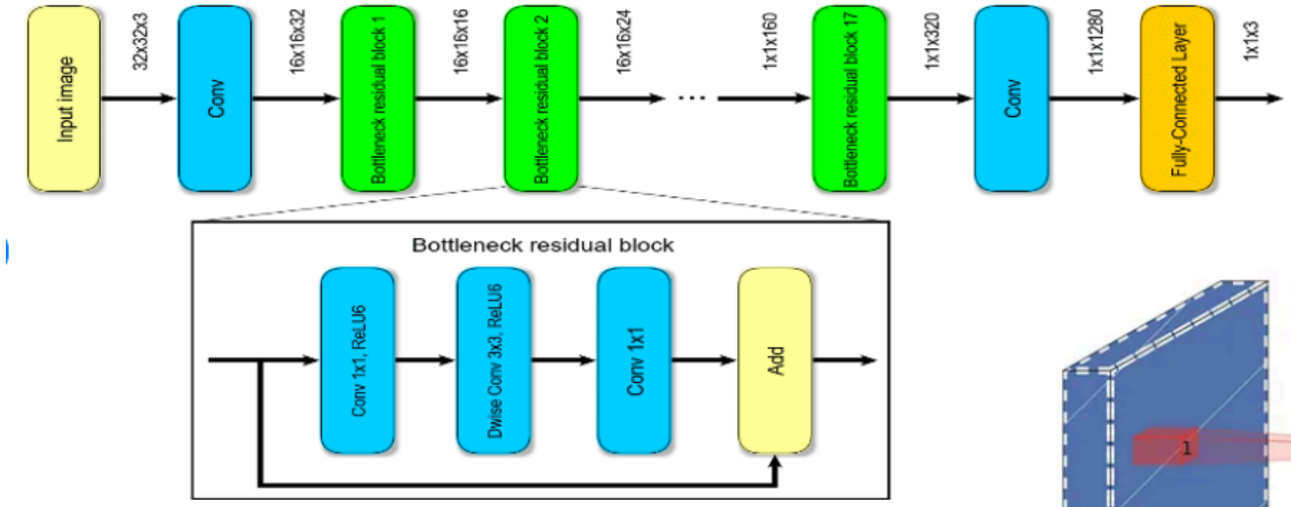


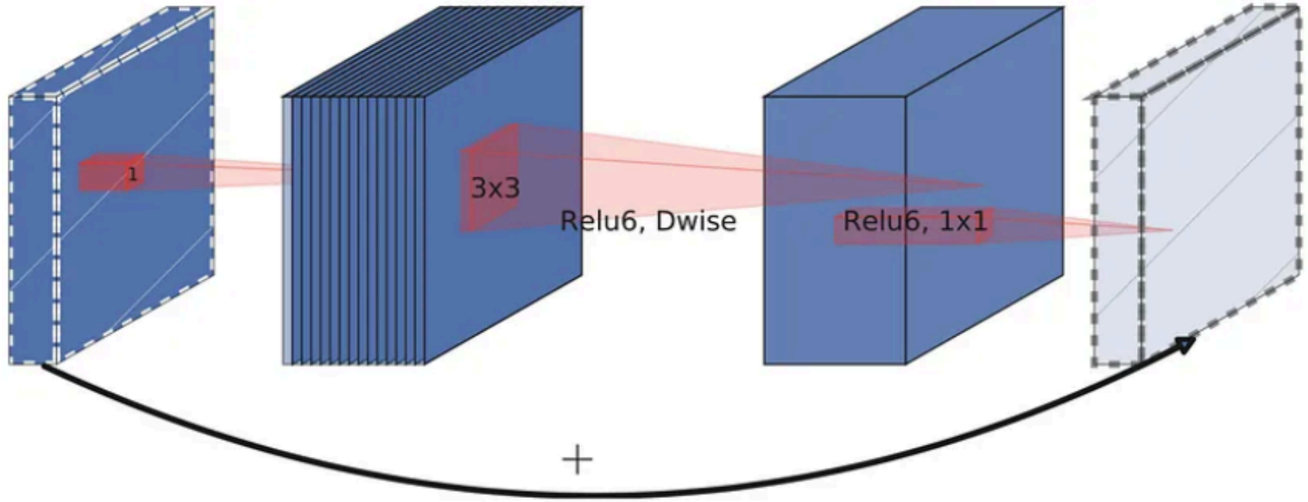
Applying a depthwise convolutional filter on 10x10x3 input volume outputs 8x8x3 volume



MOBILENET V3 ARCHITECTURE

- We trained state-of-the-art Mobilenet V3 after performing transfer learning (last 5 layers are left to be trainable)
- we got a model size of 9MB and accuracy of 76.67 and 5 Million parameters
- We added a GAP layer to reduce the dimensionality of the output recieved from pretrained model , [3]



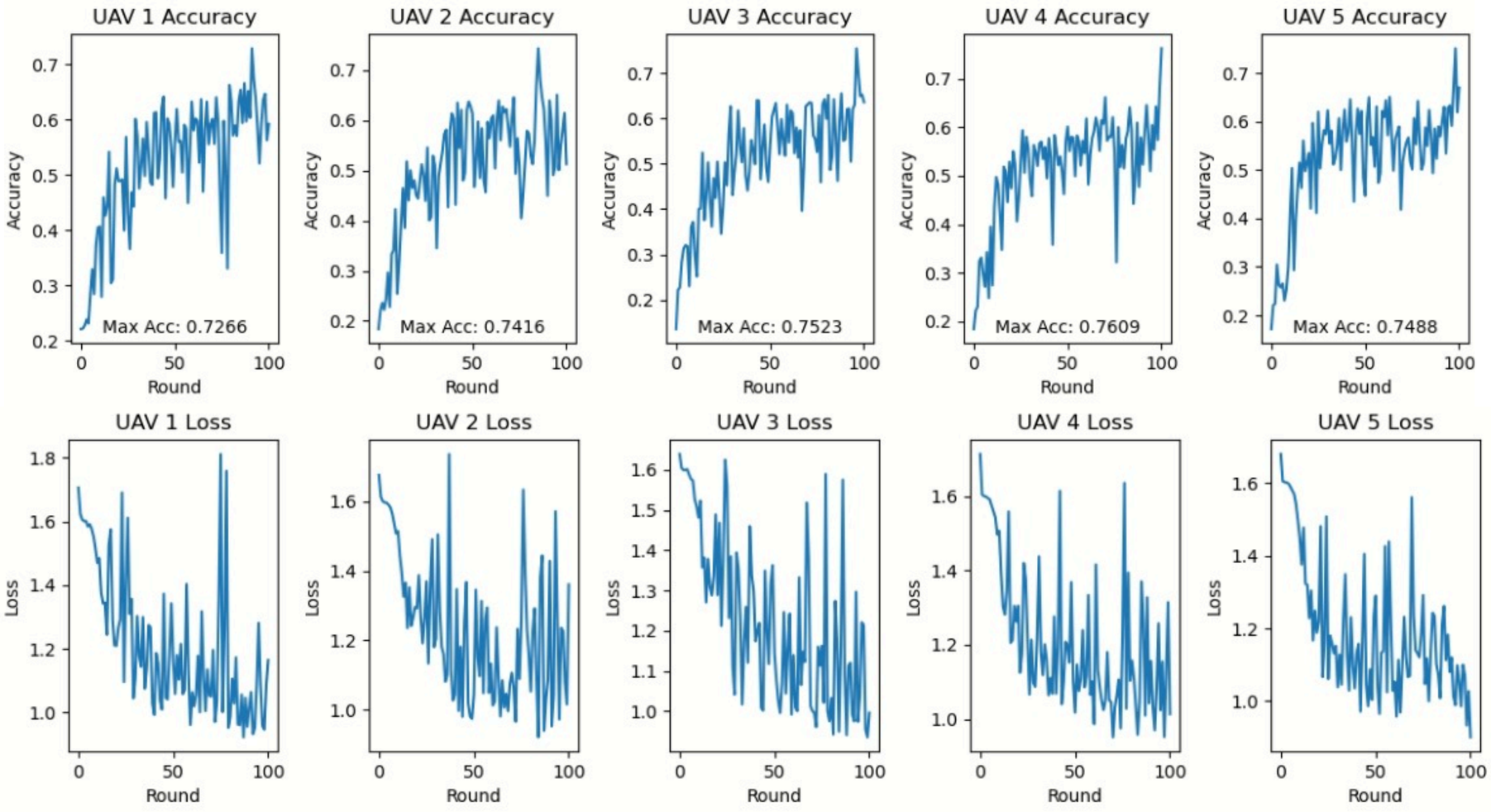


MOBILENET V3 ARCHITECTURE

- MobileNetV1 introduced the basic idea of making things efficient, Introduced the basic idea of depthwise separable convolutions
- MobileNetV2 added "inverted residuals" (they expand the number of channels before the depthwise convolution and then compress them back afterwards.) and "linear bottlenecks",
- MobileNetV3, adds an attention mechanism that adaptively recalibrates channel-wise feature responses.
- "Squeeze" operation: Global average pooling to get channel-wise statistics.
- "Excitation" operation: Two small fully-connected layers to produce channel-wise scaling factors.
- NAS is an automated technique used in the development of MobileNetV3, NAS searches through a predefined space of possible network architectures to find optimal configurations.

RESULTS/OUTPUTS

- Now we trained state-of-the-art Mobilenet V3 after performing transfer learning on a Sparse Mesh Network.
- we got F1 Score of 72.39% and a model size of 32MB and max-min accuracy of 74.32 and 50,000 parameters.
- We added a GAP layer to reduce the dimensionality of the output recieved from pretrained model.



DenseNet

- DenseNet is designed to maximize feature reuse thereby enhancing the model’s efficiency and accuracy.
- Traditional CNNs pass information from one layer to the next sequentially. DenseNet changes this by connecting each layer to every other layer, ensuring the flow of information across the entire network.
- Traditional CNNs may end up learning similar or redundant features at different layers. DenseNet avoids this by making earlier features directly available to later layers, which reduces redundancy.
- DenseNet, with its feature reuse mechanism, requires fewer parameters, this makes DenseNet more parameter-efficient, meaning it can achieve high accuracy with fewer parameters, which leads to faster training.

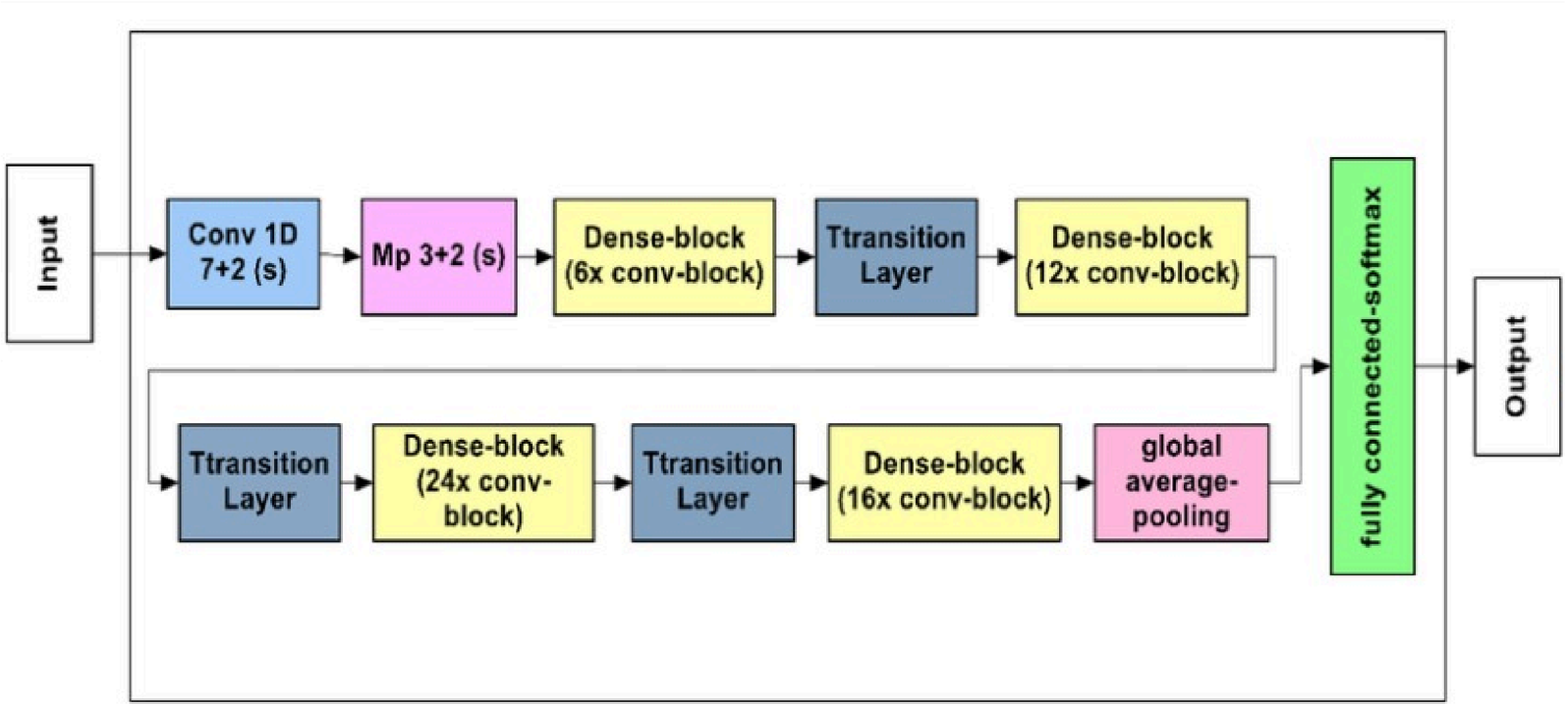
DenseNet121 Architecture

- DenseNet121 consists of 121 layers in total.
- DenseNet121 is composed of 1 Initial Layer, 4 Dense Blocks, 3 Transition Layers, and 1 Final Layer(Fully connected layer).
- The Initial layer takes input images of size 224x224 pixels with three color channels (RGB). It consist of a large convolutional layer followed by a max-pooling layer.It extract basic features and reduce the spatial dimensions of the input, preparing the image for deeper feature extraction.
- Dense Blocks are the core of the DenseNet architecture. They consist of several layers that are densely connected with in the block and each layer is composed of 1x1 Convolution which reduces the number of feature maps, 3x3 Convolution which extracts complex features.
- Transition Layers are placed between dense blocks. They contain 1x1 Convolution which reduces the number of feature maps and 2x2 Average Pooling which reduces the spatial dimensions thereby reducing the size of the model.

DenseNet121 Architecture cont..

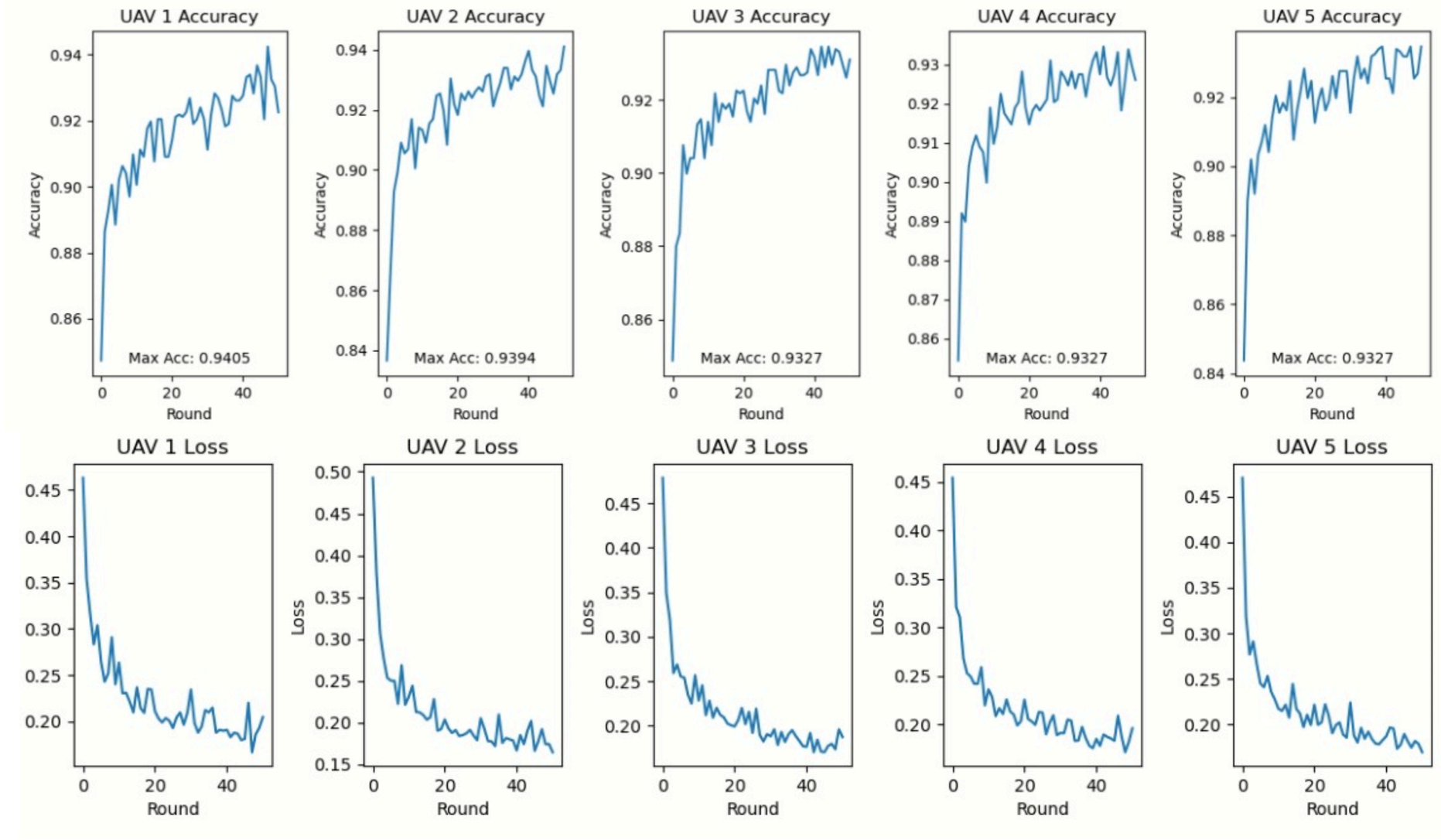
- The Final Layers of DenseNet121 consist of a global average pooling layer which reduces each feature map into a single value. It reduces the size of the data passed to the fully connected layer.
- Fully connected layer maps the pooled features to the output classes for classification.

Architecture Flow



Results/Outputs

- Now we trained DenseNet121 after performing transfer learning on a Sparse Mesh Network.
- we got F1 Score of 92.64% and a model size of 32 MB and max-min accuracy of 94.24 and 79,78,856 parameters.
- We added a GAP layer to reduce the dimensionality of the output received from pretrained model.



COMPARISIONS

Model Name	Accuracy	Parameters	Model Size
MobileNet V2	97.01%	7,26,405	9MB
ResNet 50	72.15%	5,88,97,030	224 MB
VGG 16	97.60%	1,47,17,253	54MB
MobileNet V3	74.32%	50,00,000	32MB
DenseNet 121	94.24%	79,78,856	32MB

Priority Based Fed Avg.

- So, Everywhere we are averaging for aggregation.
- But we think that it has a great impact on the overall models performance
- Because at last, we will be updating the model based on the weights we recieved
- So Why dont we add priority to weights
- assign less weight to the outliers, hence our resultant model weights wont take too much effect from the Outliers

Priority Based Fed Avg Contd.

$$MW_{\text{out}} = \frac{\sum_{i=1}^{N}(mw_i \cdot p_i)}{\text{total\_nodes}}$$

- mw_i denotes weight of model in ith node
- p_i denotes priority assigned to the weights of model in “ith” node
- the priorities can be assigned based on the outlier analysis, say for extreme outliers which are out of 95% then we can assign very least priority

References

[1]H. Du et al., “Decentralized Federated Learning Strategy with Image Classification using ResNet Architecture,” 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Jan. 2023, doi: 10.1109/ccnc51644.2023.10060275.

[2]Y. Qu et al., “Decentralized Federated Learning for UAV Networks: Architecture, Challenges, and Opportunities,” IEEE Network, vol. 35, no. 6, pp. 156–162, Nov. 2021, doi: 10.1109/mnet.001.2100253.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, doi: 10.1109/cvpr.2018.00474.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, doi: 10.1109/cvpr.2016.90.

[5] Srikanth Tammina, International Journal of Scientific and Research Publications, Volume 9, Issue 10, October 2019, doi: 10.29322/IJSRP.9.10.2019.p9420

Thank You