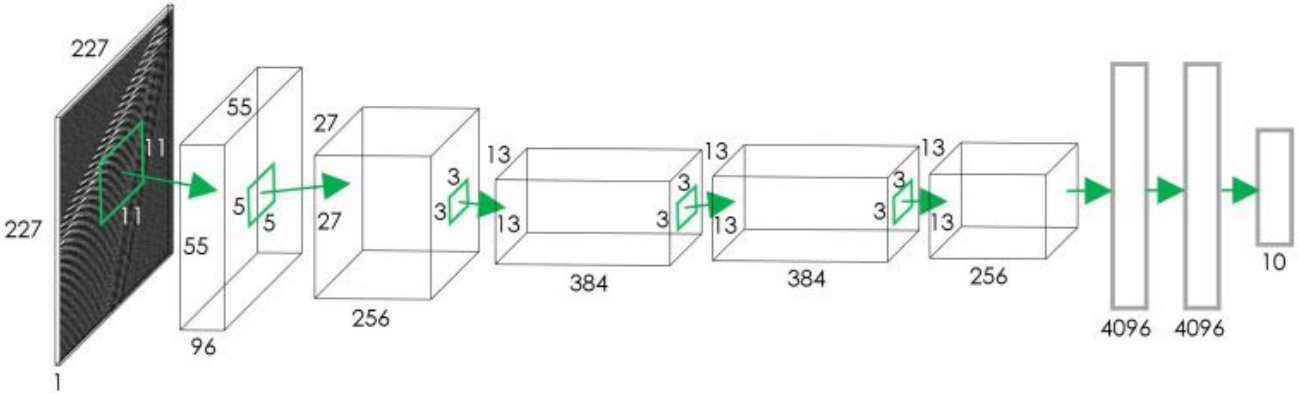
Embedding Augmentation and Hybrid Convolutional Transformers for Ship Wake Classification

Siddharth Sandu



Introduction

*Current SOTA AlexNet (2012) from the paper [1]

- Importance of vessel type classification in SAR imagery for maritime applications
- Limitations in using real SAR imagery for deep learning classification due to data scarcity and labor-intensive labeling
- Extreme Noise levels makes it difficult to leverage traditional Computer Vision Algorithms.
- Existing work does not leverage modern deep learning algorithms with the current SOTA being MobileNetv2 [2] and AlexNet [1] [3].
- Work can also be done to address data scarcity and data efficient ML.

Primary Research Questions

1. Does Denoising an Image improve Deep Learning Performance?
2. Are Transformers and Hybrid Transformer models better than conventional Convolutional Neural Networks?
    a. Conventional Models: MobileNetV2, ResNet, DenseNet.
    b. Vision Transformers: ViT, Hierarchical Transformers: ConvNext
3. Does Augmenting in the latent space enhance a model's performance?
    a. E-Mixup
    b. E-Stitchup

Dataset

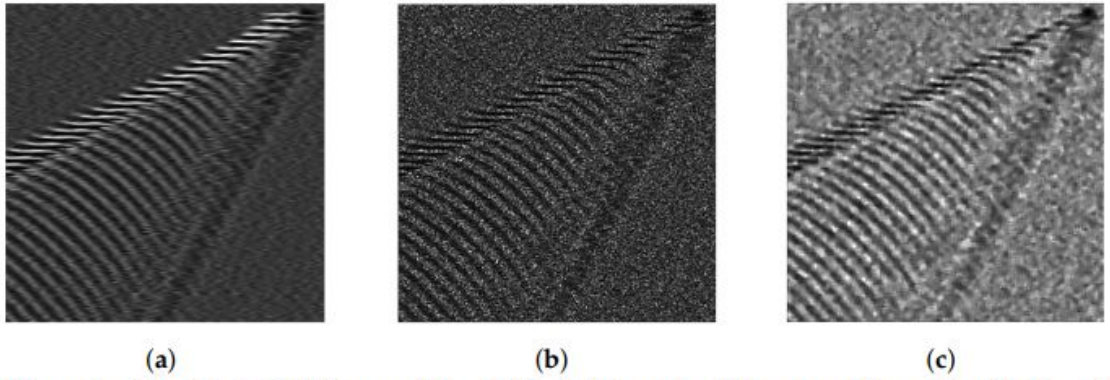
The parameters that are used for simulating the Ship wake images in the paper [1]. They have made the processed images publicly available.

Ship Type	Length, m	Beam, m	Draft, m	Velocity, m/s					
				V_{s1}	V_{s2}	V_{s3}	V_{s4}	V_{s5}	V_{s6}
Cargo I	195	26	7.1	5	6.2	7.4	8.6	9.8	11
Cargo II	366	51	13.6	5	6.4	7.8	9.2	10.6	12
Tanker I	108	17	5.6	5	6	7	8	9	10
Tanker II	228	32	11	5	5.8	6.6	7.4	8.2	9
Passenger Vessel I	86	18	2.5	5	5.6	6.2	6.8	7.4	8
Passenger Vessel II	186	28	6.5	5	6.6	8.2	9.8	11.4	13
High Speed Craft I	100	17	2.5	5	7.8	10.6	13.4	16.2	19
High Speed Craft II	31	7	3.8	5	8	11	14	17	20
Fishing Vessel I	70	16	8	5	5.8	6.6	7.4	8.2	9
Fishing Vessel II	27	8	5.1	5	5.4	5.8	6.2	6.6	7

- We leverage SynthWakeSAR [1], a publicly available dataset for classification of vessel types in SAR imagery.
- The original dataset contains of 46,080 images across 10 classes and were generated for three different distributions
- The images were simulated using AssenSAR [4] [5] and have been converted to grayscale images.
- We combine the subclasses to get 5 distinct classes and make the experiment computationally feasible.

Dataset (Continued)

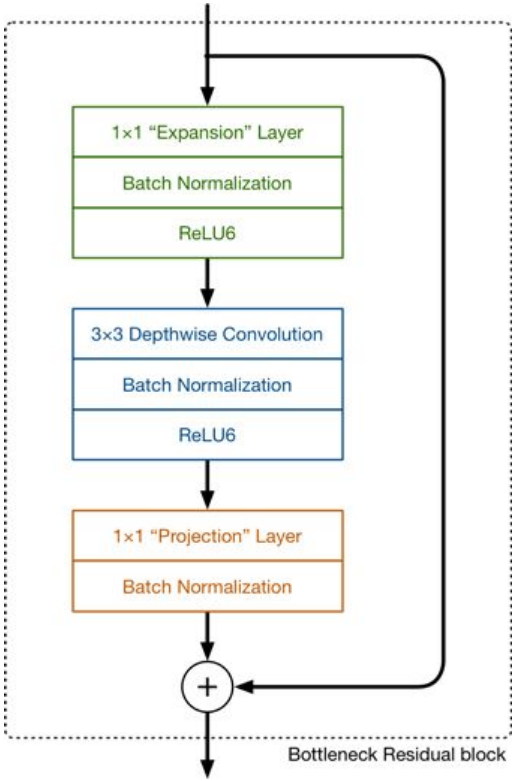
Simulated SAR images (θr = 20◦) of ship wake (Passenger Vessel I) with Vs = 8 m/s, Ds = 45◦ and Vw = 3 m/s: (a) noise-free I; (b) with noise In; (c) denoised Id [1].



- As speckle noise is key for any SAR image, we also test the models with Noisy (N), and Denoised Images (ND).
- We sample a total of 2750 images, 550 images per class, and the same images are sampled for the Noise-Free (NF) and the other two distributions.
- We finally resize them to a size of 128x128 and save them as RGB arrays to facilitate the model configurations.

MobileNetV2

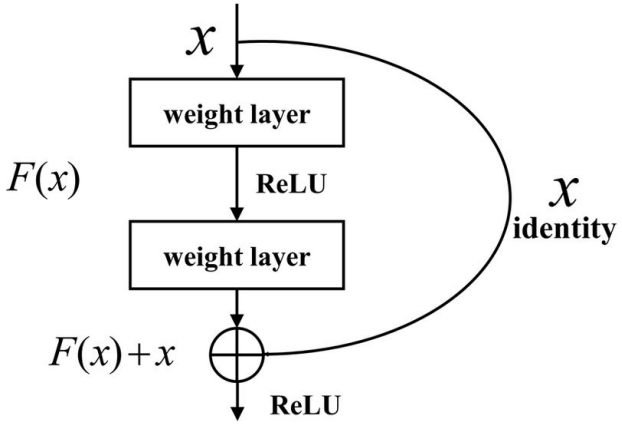
- MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet [6].
- In Version 2 (V2) of the convolutional block design, three layers are employed: depthwise convolution, projection layer, and expansion layer [6].
- The projection layer reduces the number of channels, acting as a bottleneck, while the expansion layer increases channel count, facilitating data transformation before depthwise convolution [6].



*Main building block for MobileNetV2 [6]

ResNet50

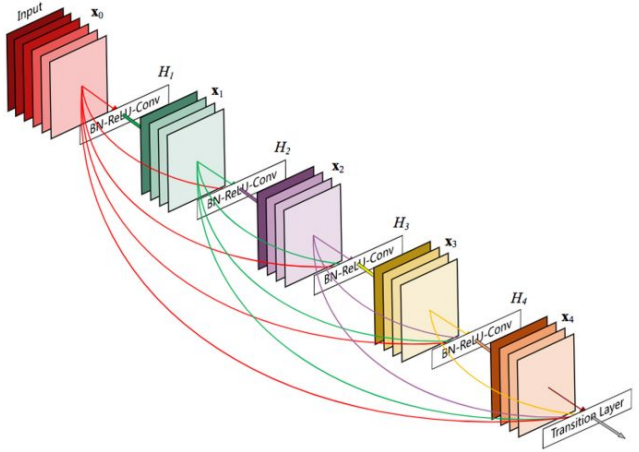
- 50-layer deep convolutional network for high-accuracy image recognition.
- Mitigates vanishing gradients with residual blocks and skip connections.
- Features 1x1, 3x3 convolutions, and efficiency via bottleneck design.
- Maintains performance despite depth, excelling in image classification.
- Pioneered advancements in deep learning, addressing core challenges.



*Main building block for a Residual Network [7]

DenseNet121

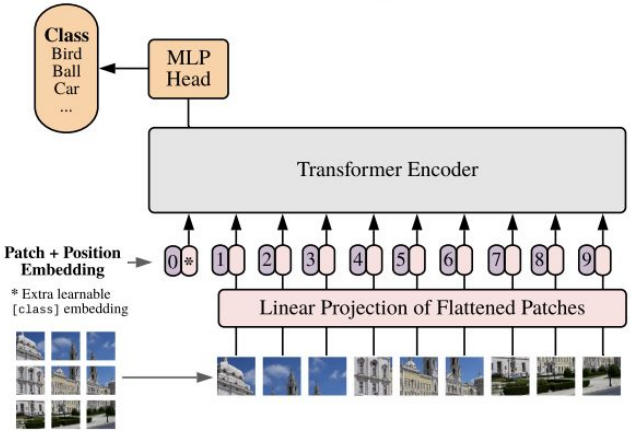
- A 121-layer network with dense connectivity for image recognition.
- Enhances feature reuse, reducing vanishing gradient issues.
- Connects each layer to every other layer directly.
- Efficient through feature concatenation, not addition.
- Demonstrated superior accuracy in fewer parameters.
- Influential in deep learning for robust feature learning.



*Structure of a Dense Block [8], which is repeated throughout a model.

Vision Transformer (ViT)

- **Attention Mechanism for Vision**: Vision Transformers (ViTs) apply the transformer architecture, originally designed for sequential data like text, to image data. Instead of using convolutional layers like traditional CNNs, ViTs rely entirely on self-attention mechanisms to capture global and local dependencies within the image. This allows ViTs to process images as sequences of patches, enabling them to learn relationships between different parts of the image without explicitly modeling spatial hierarchies.
- **Patch Embeddings**: ViTs divide the input image into a grid of fixed-size patches, which are then linearly embedded into high-dimensional vectors. These patch embeddings serve as the input to the transformer encoder, where self-attention mechanisms are applied to capture relationships between patches. By processing image patches as sequential data, ViTs can effectively capture both local and global context, leading to strong performance on a variety of vision tasks.

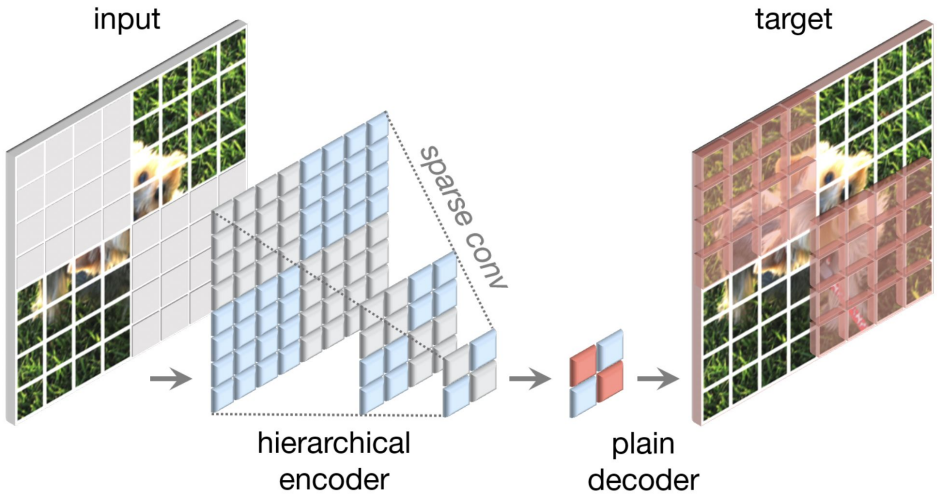


*Overview of the ViT architecture [9]

Here, we use a shallower version with the following parameters to accommodate the available computational resources, patch_size=16, num_layers=12, activation='linear', hidden_size =144, num_heads = 12 mlp_dim = 512

ConvNext

*Overview for the ConvNext Model [10]

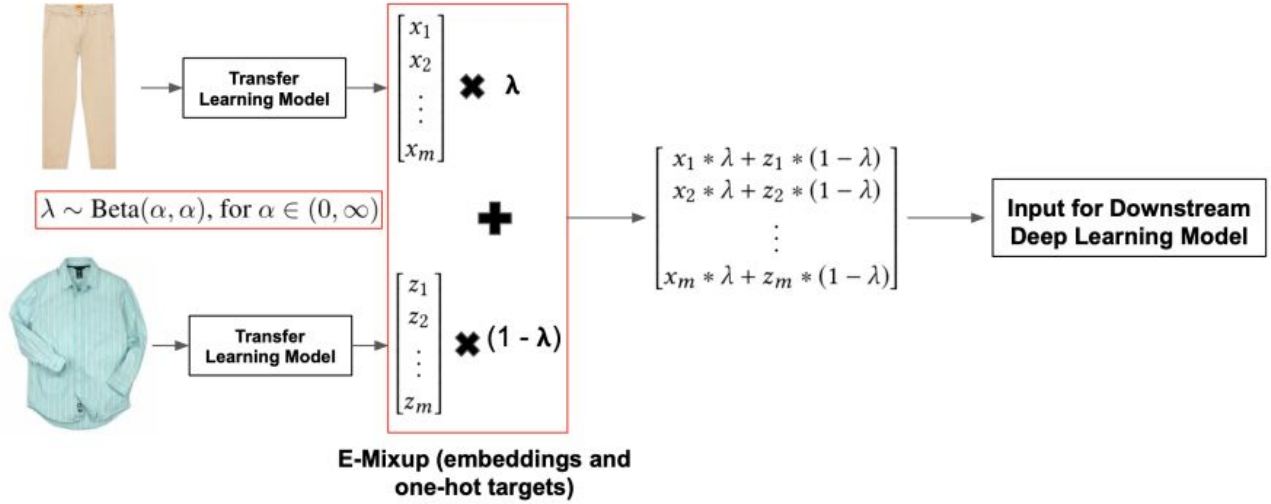


- ConvNext introduces a hybrid architecture that combines the strengths of convolutional neural networks (CNNs) and transformer models [10].
- Unlike conventional CNNs, which rely solely on convolutional layers, ConvNext integrates transformer blocks to capture long-range dependencies in addition to local features.
- This hybrid approach enhances the model's ability to understand both global and local contexts within the input data [10].
- We leverage the ConvNeXtTiny model which is the most resource efficient model in this family to train.

- ConvNext employs convolutional layers to extract hierarchical features from the input data, preserving spatial relationships and local patterns [10].
- These features are then fed into transformer blocks, where self-attention mechanisms facilitate the learning of contextual dependencies across different levels of abstraction.
- This hierarchical representation enables ConvNext to effectively capture both fine-grained details and high-level semantics in the input data [10].

E-Mixup

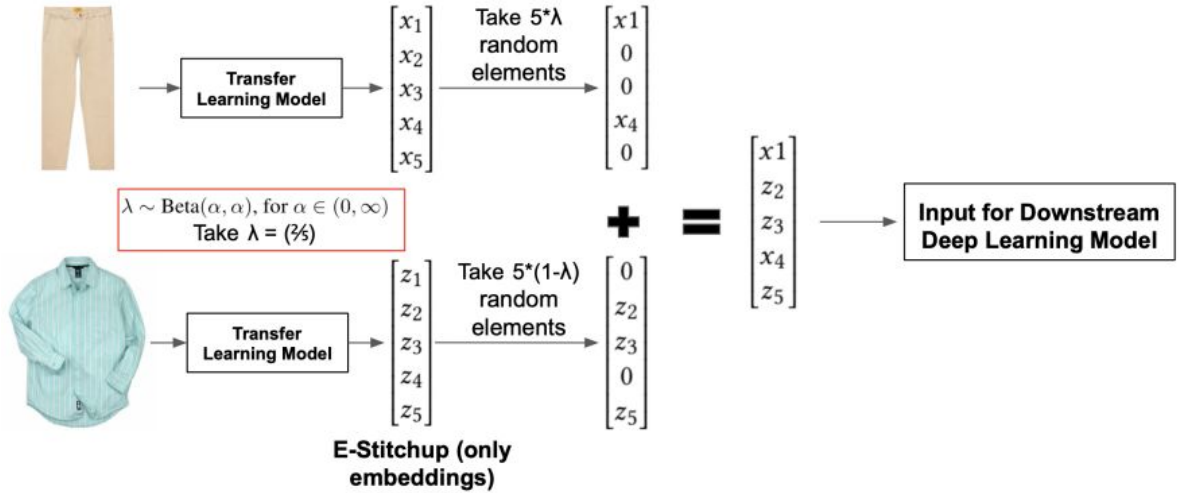
*Both the embedding inputs and their associated target vectors (i.e., one-hot prediction targets) are augmented using the same process outlined above



- E-Mixup is introduced as a form of data augmentation, inspired by Mixup but operating on embeddings rather than raw images. It involves sampling a random value, lambda, from a Beta distribution with a parameter alpha [11].
- E-Mixup then takes a weighted average over the embedding inputs of two unique training examples, with lambda serving as the weight. This process is applied to both input embeddings and their associated target vectors, replacing one-hot targets with the weighted average of the two labels [11].

E-Stitchup

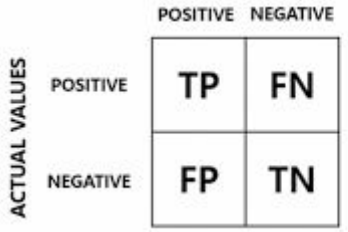
*The label vectors associated with these embedding inputs are not handled in the same way, but are instead handled by taking a weighted average of the two label vectors, as in E-Mixup



- E-Stitchup, a variation of Mixup data augmentation, draws inspiration from experiments in which Mixup involves randomly pasting cropped sections of one image over another instead of averaging pixels [11].
- In E-Stitchup, two embeddings are combined by randomly selecting values from each original embedding, determined by a sampled value, lambda, from a beta distribution. This process effectively blends information from different embeddings, contributing to improved model generalization [11].

Experimental Setup

- We generate the training and testing sets through an 80-20 train test split and the same random split is used to accommodate the three distributions.
- All models are subjected to the same arrays, a robust Early Stopping criteria and a ReduceLROnPlateau strategy for a maximum of 25 epochs, as through initial testing, we were able to observe sufficient convergence.
- The modality of the experiments were performed using the Google Colaboratory T4 GPU Instance.
- We also assess these models through t-SNE plots and analyze them for a multitude of ML metrics.



$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

LE Generator

```
def create_resnet50_base(input_shape):
    base_model = ResNet50(include_top=False,
input_shape=input_shape)
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(125, activation='sigmoid')(x)
    model = Model(inputs=base_model.input, outputs=x)
    return model

def create_classification_model(base_model, num_classes):
    for layer in base_model.layers:
        layer.trainable = True

    x = Dense(num_classes,
activation='softmax')(base_model.output)
    model = Model(inputs=base_model.input, outputs=x)

    return model
```

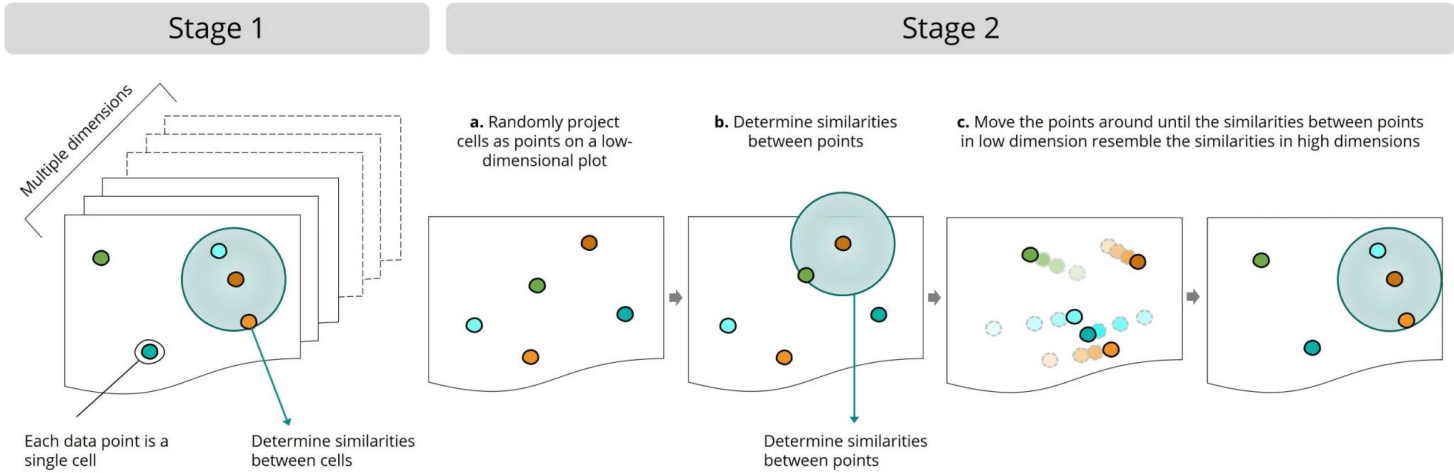
Classification Head

```
Classifier_model = Sequential()
Classifier_model.Input = (125,)

Classifier_model.add(Dense(512,activation='relu'))
Classifier_model.add(Dense(512,activation='relu'))

Classifier_model.add(Dense(5, activation='softmax'))
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
Classifier_model.compile(optimizer=opt,loss='categorical_crosse
ntropy',metrics=['accuracy'])
```

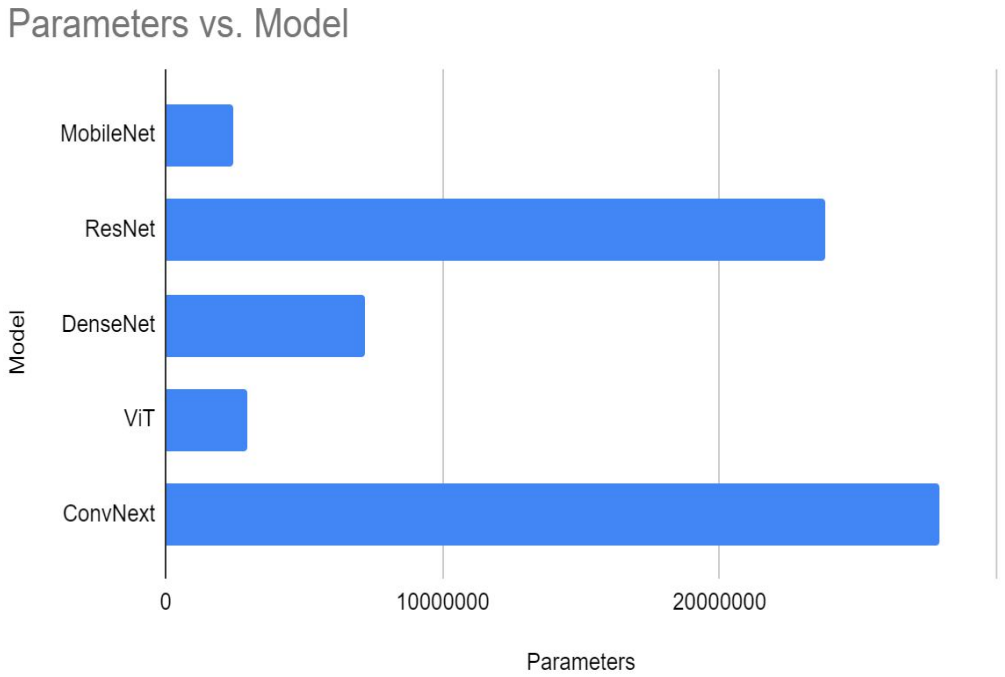
t-distributed stochastic neighbor embedding



- Neighbor embedding specifically focuses on capturing similarities in gene expression among cells. In contrast, t-SNE prioritizes the preservation of local structures. Essentially, it aims to ensure that a data point retains similar neighboring points (i.e., cells with similar expression profiles) in the lower-dimensional representation as it did in the original high-dimensional space [12].
- The term "stochastic" is synonymous with "random" and highlights the random nature of the algorithm's projection of cells onto a low-dimensional plot during the second stage of the process. Because of this stochastic aspect, each iteration of a t-SNE plot may yield different results [12].
- The designation "t-distributed" pertains to the mathematical method used to determine similarities between points in the lower-dimensional dataset. Specifically, t-SNE utilizes the statistical approach of Student's t-distribution for this purpose [12].

Model Parameters

Model	Parameters
MobileNet	2418739
ResNet	23844467
DenseNet	7166259
ViT	2925779
ConvNext	27916883



Results (NF)

Model	Accuracy	Precision	Recall	F1-Score
MobileNet	0.19091	0.06252	0.19091	0.09127
MobileNet + EStichup	0.32364	0.35409	0.32364	0.29876
MobileNet + EMixup	0.35091	0.38317	0.35091	0.31337
ResNet	0.88909	0.89031	0.88909	0.88848
ResNet + Estichup	0.88	0.881	0.88	0.87891
ResNet +EMixup	0.89091	0.89016	0.89091	0.89003
DenseNet	0.92	0.92243	0.92	0.91999
DenseNet + EStichup	**0.93455**	**0.93533**	**0.93455**	**0.93465**
DenseNet + EMixup	0.92	0.9214	0.92	0.92032
ViT	0.18727	0.03507	0.18727	0.05908
ViT + EStichup	0.19636	0.03856	0.19636	0.06446
ViT + EMixup	0.18727	0.03507	0.18727	0.05908
ConvNext	0.92727	0.92943	0.92727	0.92778
ConvNext + EStichup	0.92182	0.92499	0.92182	0.92254
ConvNext + EMixup	0.91636	0.91918	0.91636	0.91704

Results (N)

Model	Accuracy	Precision	Recall	F1-Score
MobileNet	0.21273	0.0455	0.21273	0.07497
MobileNet + EStichup	0.38727	0.39174	0.38727	0.35581
MobileNet + EMixup	0.36727	0.40035	0.36727	0.36453
ResNet	0.7	0.70033	0.7	0.6999
ResNet + Estichup	0.69636	0.69901	0.69636	0.69643
ResNet +EMixup	0.69091	0.69232	0.69091	0.69085
DenseNet	0.76545	0.76801	0.76545	0.76584
DenseNet + EStichup	0.77091	0.77288	0.77091	0.77145
DenseNet + EMixup	0.75455	0.75472	0.75455	0.75381
ViT	0.21273	0.04525	0.21273	0.07463
ViT + EStichup	0.19818	0.03928	0.19818	0.06556
ViT + EMixup	0.18727	0.03507	0.18727	0.05908
ConvNext	0.78	0.78842	0.78	0.78265
ConvNext + EStichup	0.77818	0.78538	0.77818	0.7805
ConvNext + EMixup	**0.78545**	**0.80158**	**0.78545**	**0.79003**

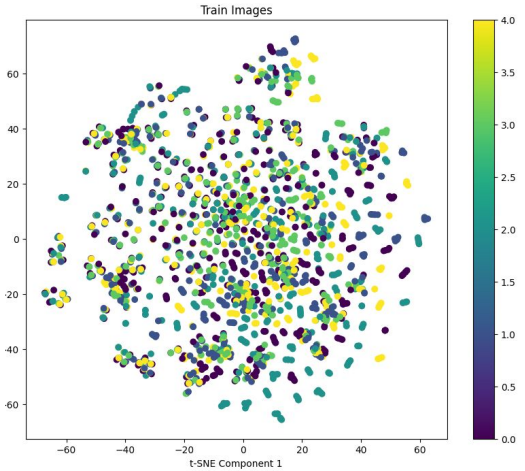
Results (ND)

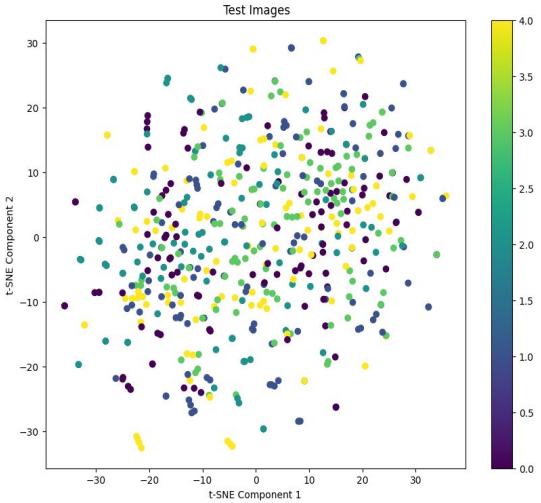
Model	Accuracy	Precision	Recall	F1-Score
MobileNet	0.21273	0.04525	0.21273	0.07463
MobileNet + EStichup	0.41273	0.36094	0.41273	0.34642
MobileNet + EMixup	0.40545	0.35856	0.40545	0.35381
ResNet	0.69636	0.69159	0.69636	0.69326
ResNet + Estichup	0.70545	0.71149	0.70545	0.70739
ResNet +EMixup	0.69455	0.68941	0.69455	0.69103
DenseNet	0.76182	0.7771	0.76182	0.76668
DenseNet + EStichup	0.76	0.77281	0.76	0.76438
DenseNet + EMixup	**0.76727**	**0.7764**	**0.76727**	**0.77075**
ViT	0.19636	0.03856	0.19636	0.06446
ViT + EStichup	0.19636	0.03856	0.19636	0.06446
ViT + EMixup	0.19818	0.03928	0.19818	0.06556
ConvNext	0.71273	0.75617	0.71273	0.71851
ConvNext + EStichup	0.69636	0.71627	0.69636	0.69956
ConvNext + EMixup	0.70909	0.72726	0.70909	0.70871

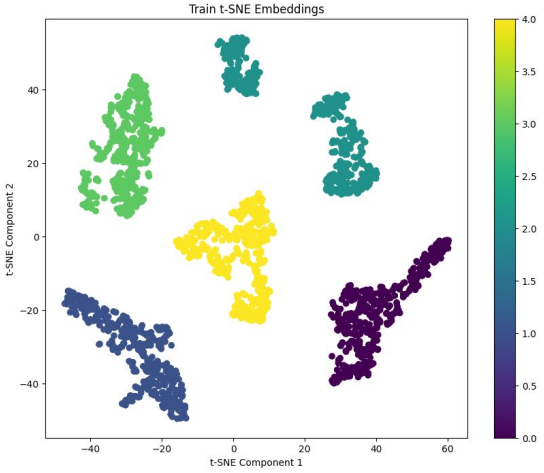
t-SNE Plots

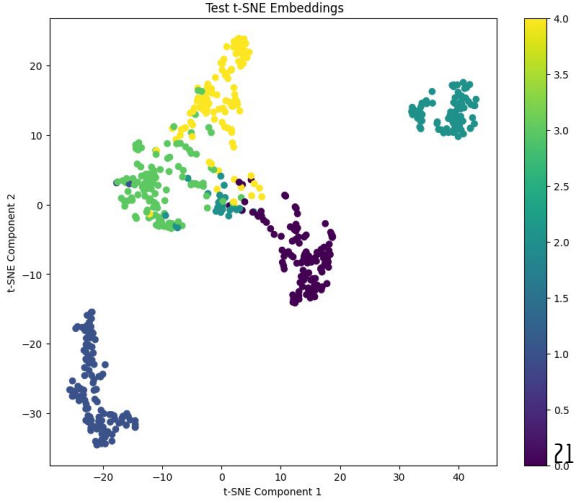
NF Best Model

DenseNet is the best performing model with an accuracy of 92% for Noise free images.



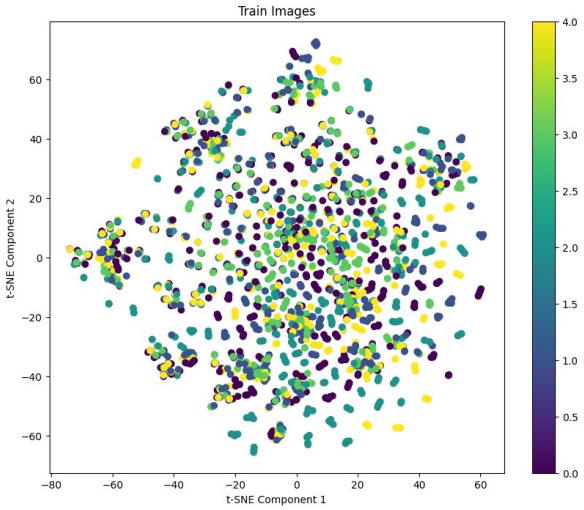


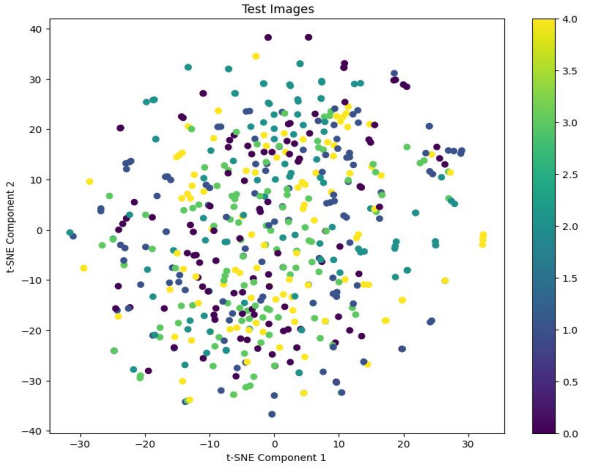


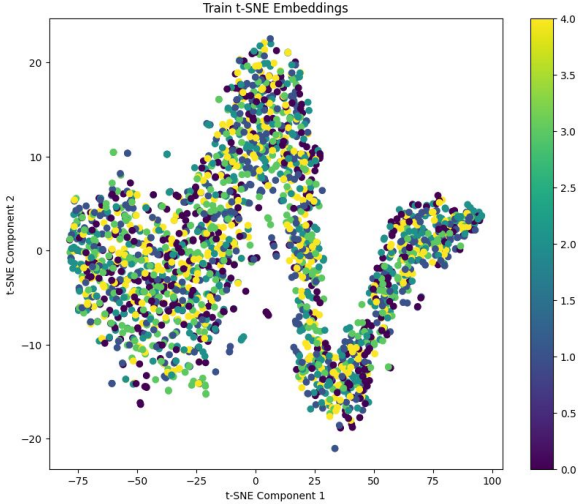


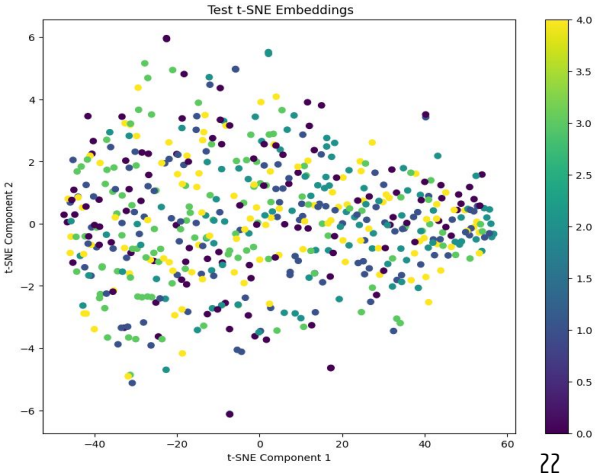
NF Worst Model

ViT is the worst performing model with an accuracy of 18.7% for Noise Free Images.



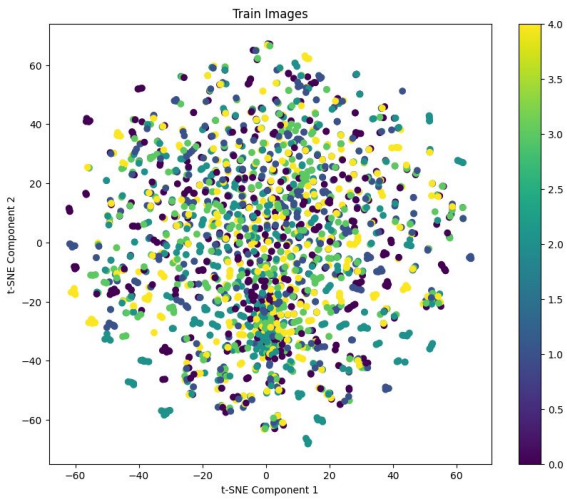


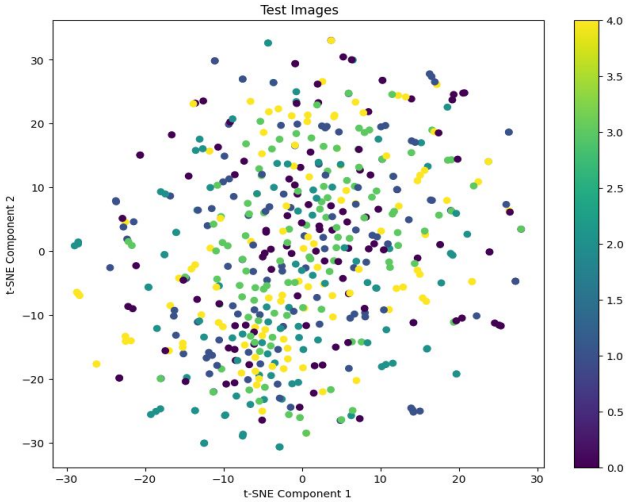


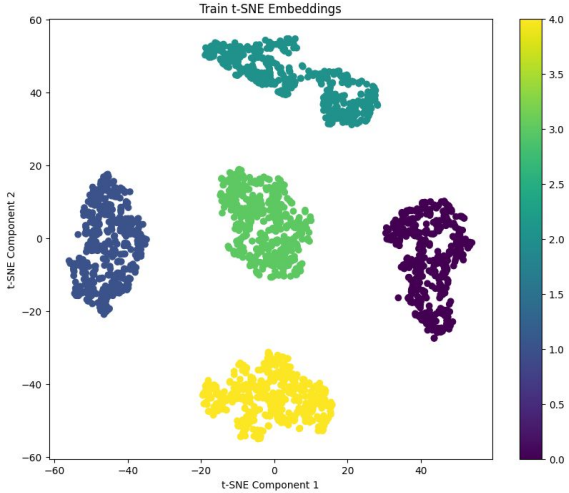


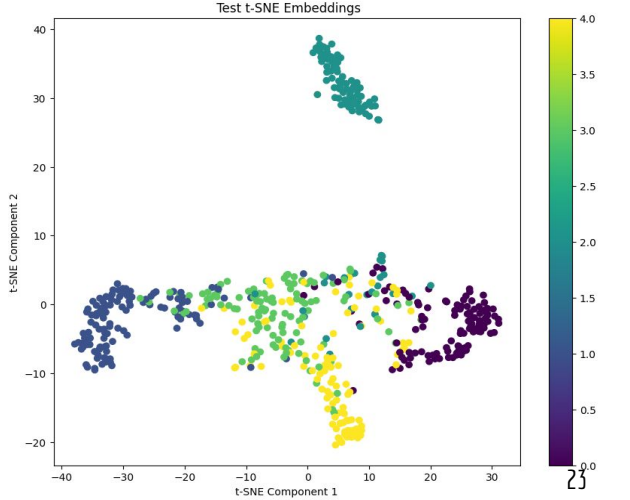
N Best Model

ConVNext is the best performing model with an accuracy of 78.5% for Noisy images.



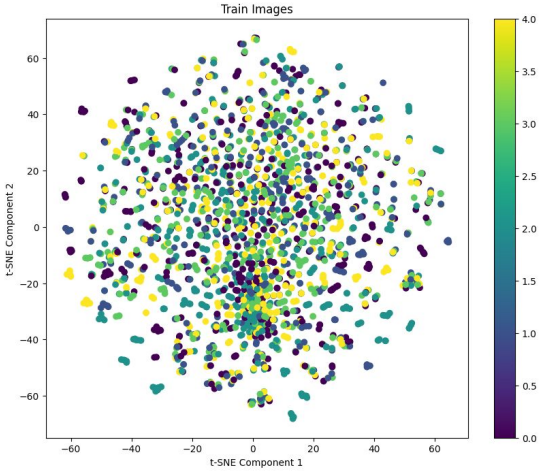


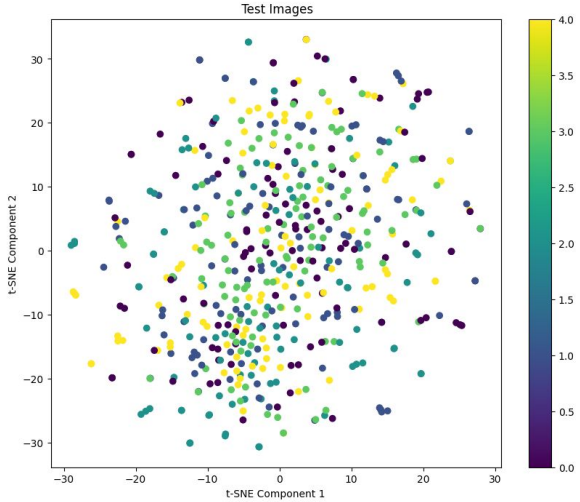


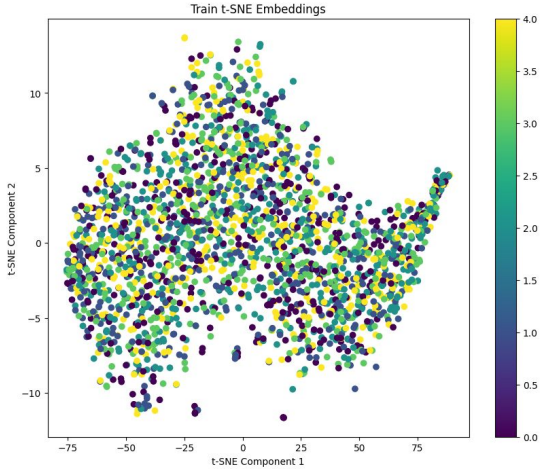


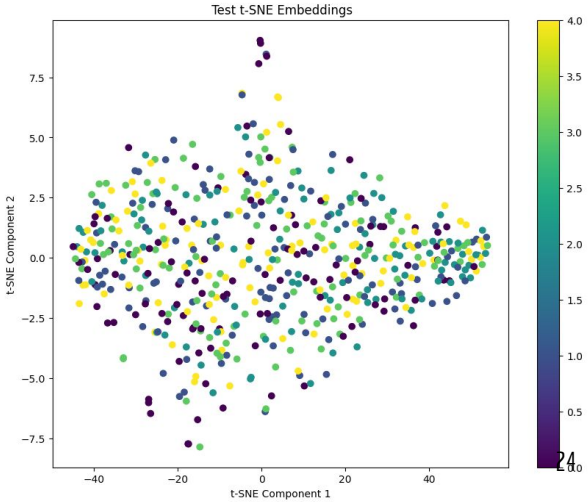
N Worst Model

ViT is the worst performing model with an accuracy of 21.2% for Noisy Images.



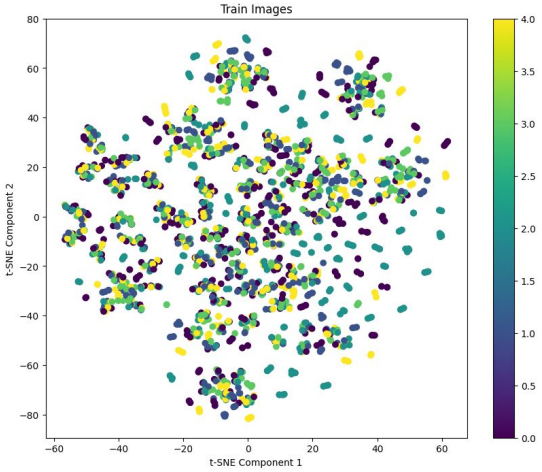


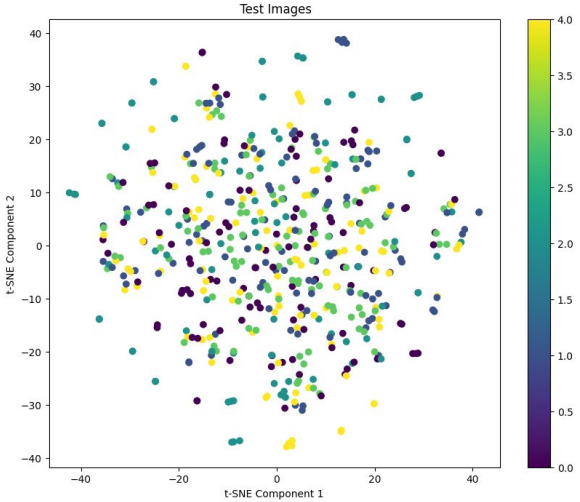


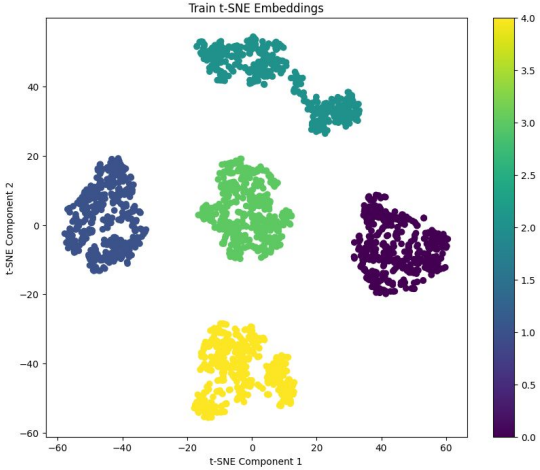


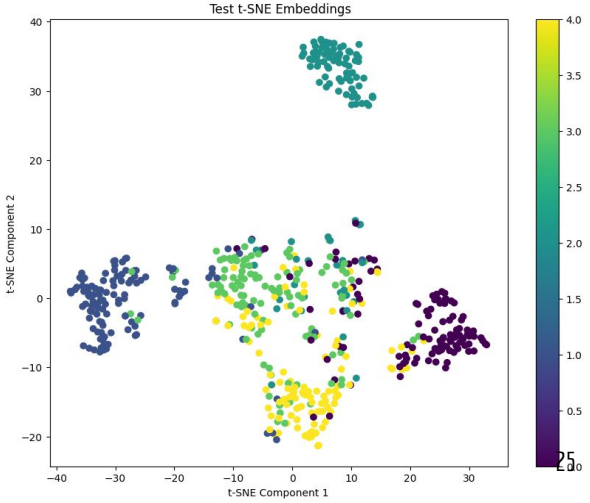
ND Best Model

DenseNet is the best performing model with an accuracy of 76.7% for Denoised images.



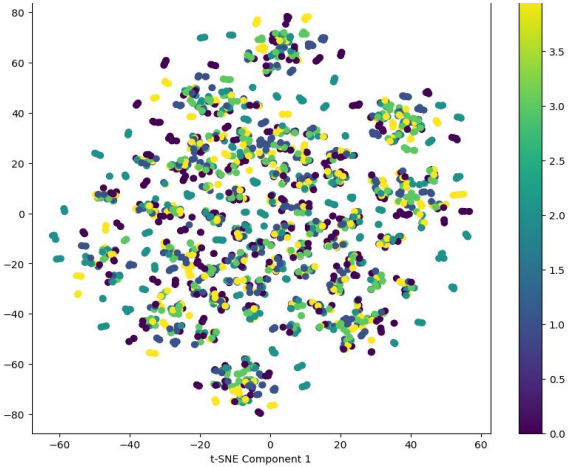


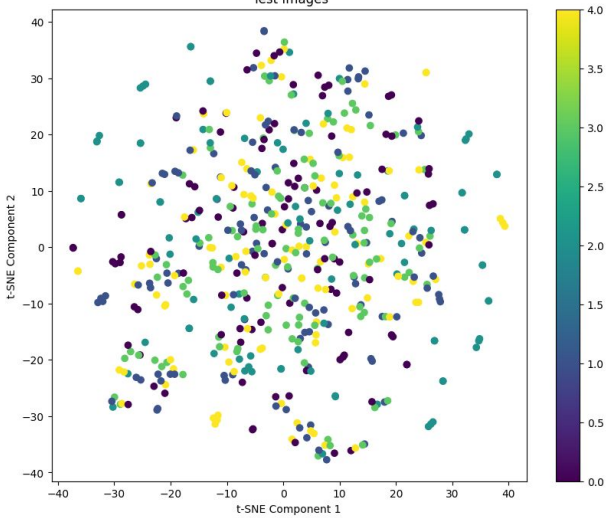


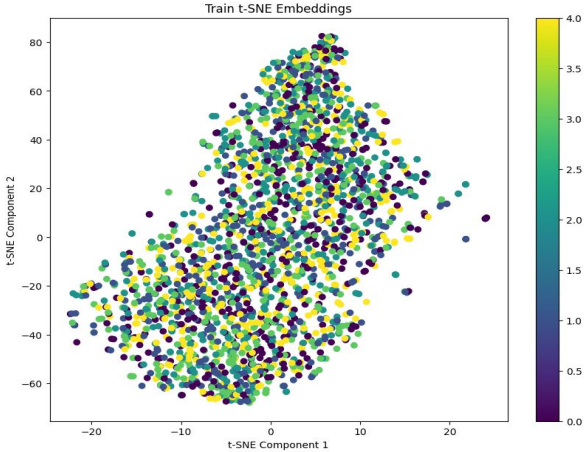


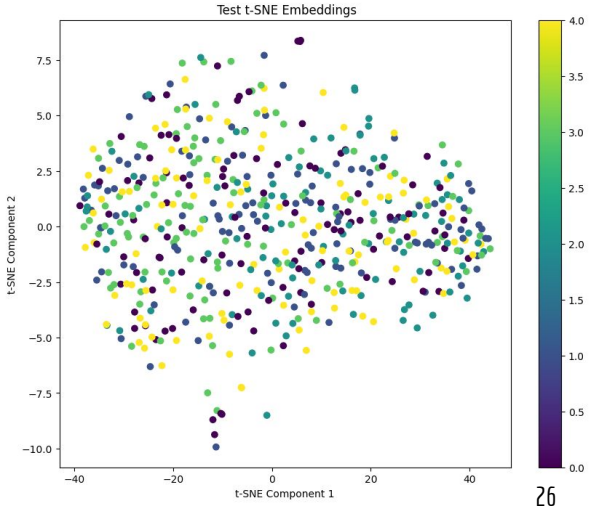
ND Worst Model

ViT is the worst performing model with an accuracy of 19.6% for Denoised Images.









Results and Discussions

- For the modality of the distributions a DenseNet was observed to be the best performing model, with ConvNext offering a near identical performance.
- The ViT consistently gave a poor performance when compared to the other classifiers followed by the MobileNet.
- The efficacy of the available Despeckling algorithm was observed to be very poor as we saw negligible performance improvements.
- The Embedding augmentation can be strongly proposed with an emphasis on E-Stitchup.

Conclusion and Future Work

- We can say that for data scarce SAR utilities, the performance of the performance of conventional approaches and hybrid or hierarchical transformers is nearly the same with the best model being a DenseNet for NF and N cases, and ConvNexts for ND images.
- Embedding augmentation can be strongly proposed as we are able to observe an average accuracy boost of 1.4% from the vanilla counterparts.
- The current standard denoising algorithms [1] are not usable and novel work can be done towards improving them.
- Deeper models with higher input channels might favor transformers.
- Work can also be done towards Siamese Networks as latent embeddings can be exploited for efficient classification.
- Research Gap pertaining to temporal efficiency.

Primary References

1. Rizaev, Igor G., and Alin Achim. "SynthWakeSAR: A synthetic sar dataset for deep learning classification of ships at sea." Remote Sensing 14.16 (2022): 3999.
2. Ding, Kaiyang, et al. "Towards real-time detection of ships and wakes with lightweight deep learning model in Gaofen-3 SAR images." Remote Sensing of Environment 284 (2023): 113345.
3. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).
4. Rizaev, I.G.; Karaku¸s, O.; Hogan, S.J.; Achim, A. Modeling and SAR imaging of the sea surface: A review of the state-of-the-art with simulations. ISPRS J. Photogramm. Remote Sens. 2022, 187, 120–140
5. Rizaev, I.; Achim, A. AssenSAR Image Simulator. Available online: https://doi.org/10.5523/bris.el0p94vgxjhi2224bx78actb4 (accessed on 25 January 2022)
6. Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Primary References (Continued)

7. Zhang, Chaoning, et al. "Resnet or densenet? introducing dense shortcuts to resnet." proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021.

8. Albelwi, Saleh A. "Deep architecture based on DenseNet-121 model for weather image recognition." International Journal of Advanced Computer Science and Applications 13.10 (2022).

9. Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

10. Liu, Z., et al. "ConvNet for the 2020s. arXiv." arXiv preprint arXiv:2201.03545 10 (2022).

11. Wolfe, Cameron R., and Keld T. Lundgaard. "E-stitchup: Data augmentation for pre-trained embeddings." arXiv preprint arXiv:1912.00772 (2019).

12. Dimitriadis, George, Joana P. Neto, and Adam R. Kampff. "t-SNE visualization of large-scale neural recordings." Neural computation 30.7 (2018): 1750-1774.

Thank You