Supplementary documents for WACV_2024 for paper ID: 2676

In this document we provided additional information for the paper 2676 submitted in algorithm track.

Here is the algorithm for the data transformation from raw HDF5 to PNG explained in section 3: Data and Preprocessing.

Algorithm 1 HDF5 to PNG

Require: $arr \in \mathbb{R}^{1000x1000}$

1: $F(x) \leftarrow P(Z <= x)$ for $Z \sim$ Normal distribution fitted to the values of arr
2: $arr \leftarrow arr - min(arr)$
3: $arr \leftarrow \frac{arr}{median(arr)} * 0.5$
4: $arr \leftarrow clip(arr, 0, 1)$
5: $arr \leftarrow F(arr)$ ▷ Transform the approximately normally distributed values to uniform ones
6: $arr \leftarrow clip(arr, 0, 1)$

Following are the optimization plots for the models that are explained in Table 1:

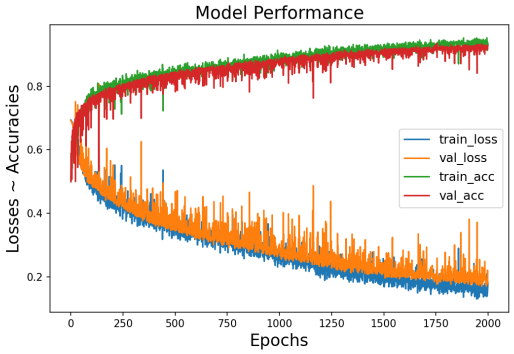


Fig: Gabor

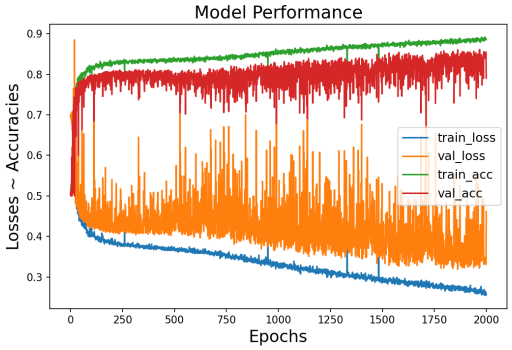


Fig: ViT

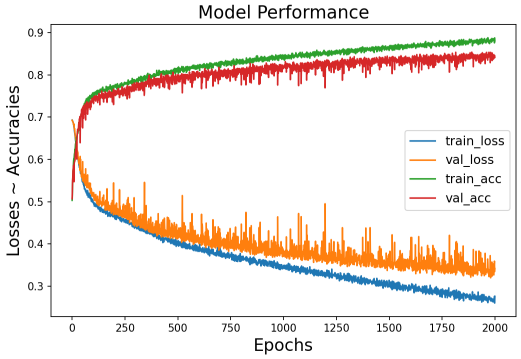


Fig: Sobel_1

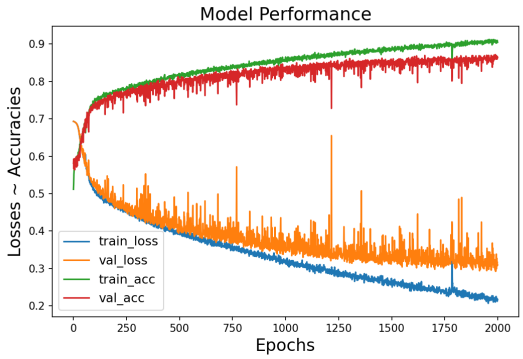


Fig: Sobel_2

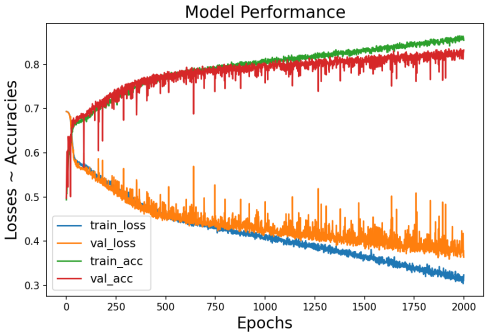


Fig: Laplacian_1

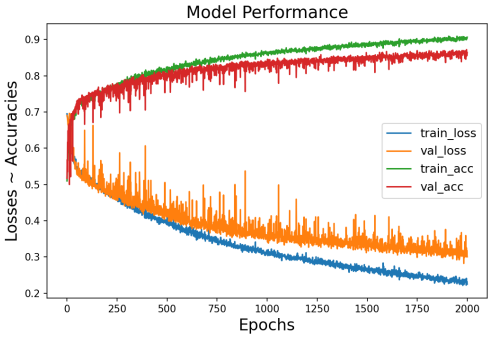


Fig: Laplacian_2

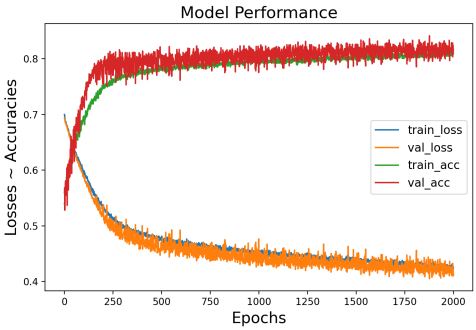
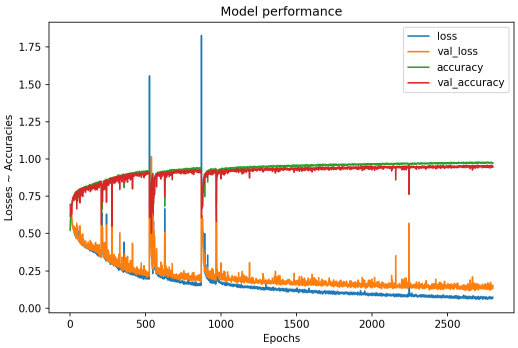


Fig: FFT_no_custom_kernel_layer



FFT: Non_trainable

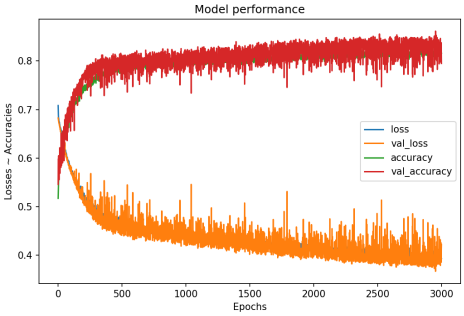


Fig: gWaveNet_noK

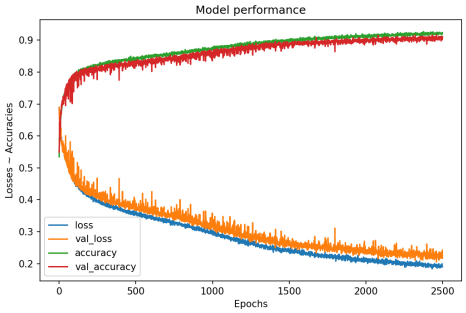


Fig: gWaveNet_64K

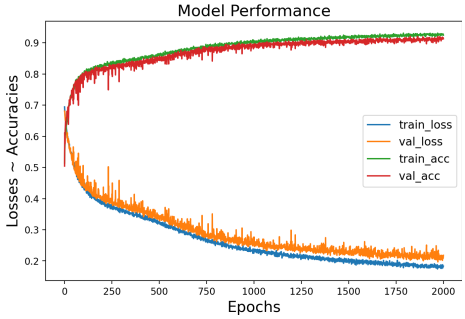


Fig: gWaveNet_3x3_non_trainable_layers

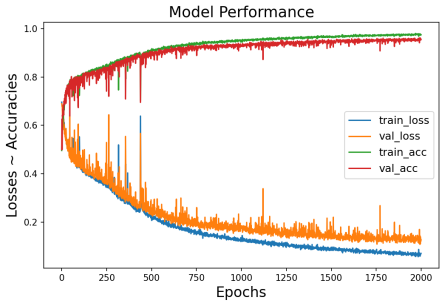


Fig: gWaveNet_5x5_non_trainable_layers

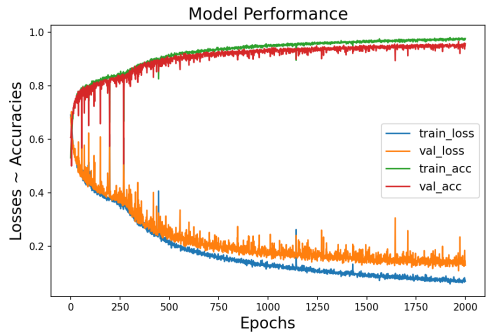


Fig: gWaveNet_9x9_non_trainable_layers

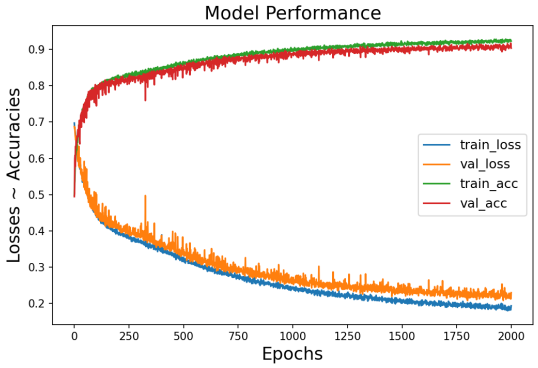


Fig: gWaveNet_9x9_non_trainable_layers

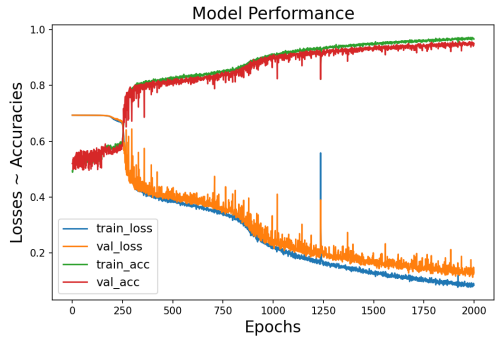


Fig: gWaveNet_3x3_kernel-applied-prior

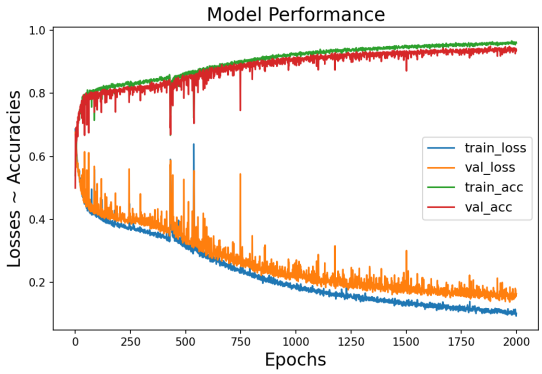


Fig: gWaveNet_5x5_kernel-applied-prior

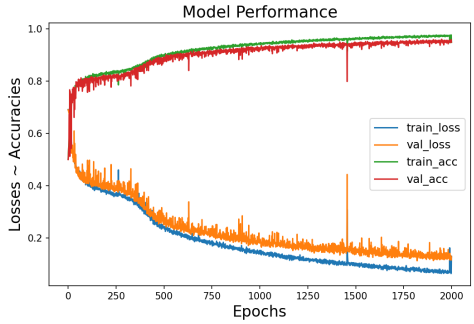


Fig: gWaveNet_7x7_kernel-applied-prior

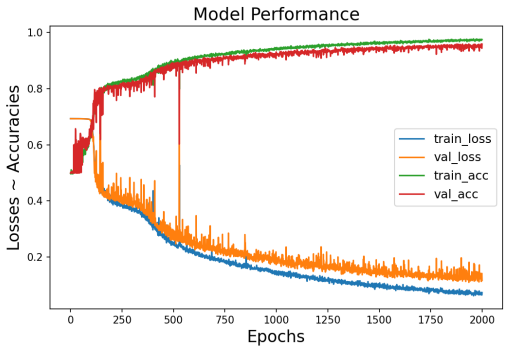


Fig: gWaveNet_9x9_kernel-applied-prior

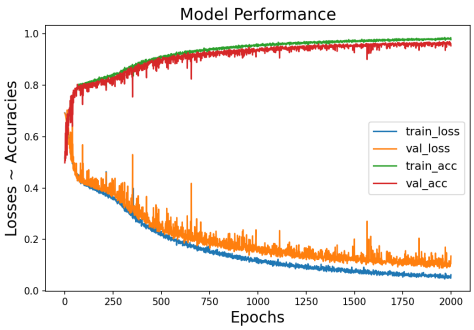


Fig: gWaveNet_3x3_trainable_kernel

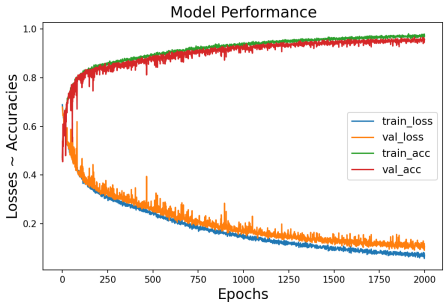


Fig: gWaveNet_5x5_trainable_kernel

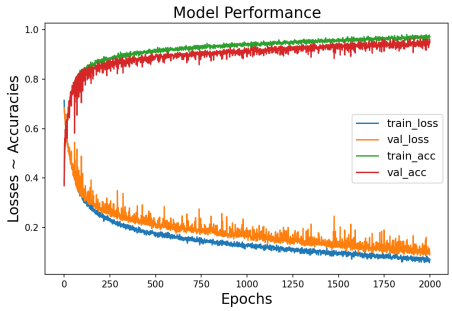


Fig: gWaveNet_7x7_trainable_kernel

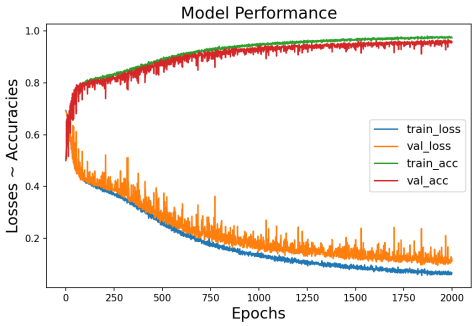


Fig: gWaveNet_9x9_trainable_kernel

Here is the AUC plot that shows the gWaveNet model is learning. We just presented one plot here as almost all of the AUC plots have a similar pattern.

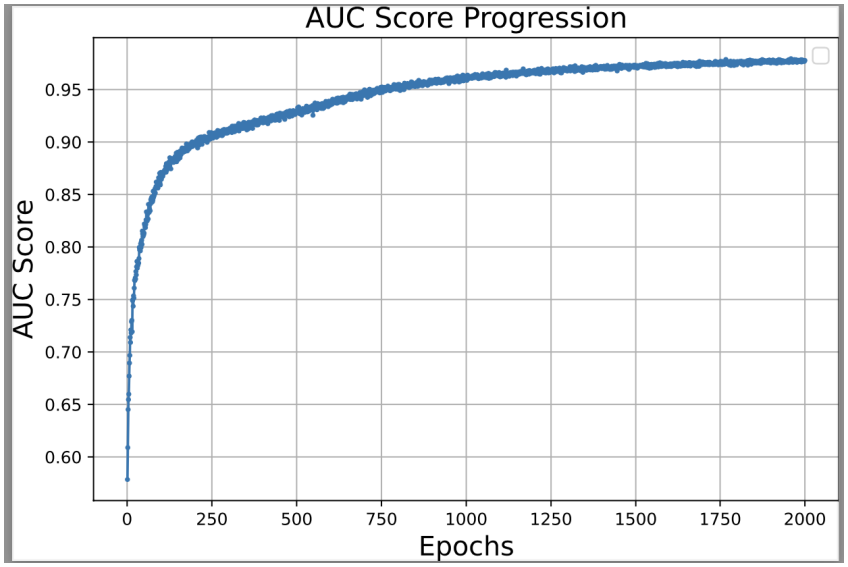


Fig: AUC score progression for gWaveNet 7x7-trainable-kernel.

Overall Observation:
Gabor Filter trained well, however, if we examine some of the images were misaligned with the filters, which indicate not enough robustness. The following block of image shows that the filters are designed to match the original images in the left column, however the output in the right column does not match the pattern. In such cases the gabor will misclassify the gravity waves.

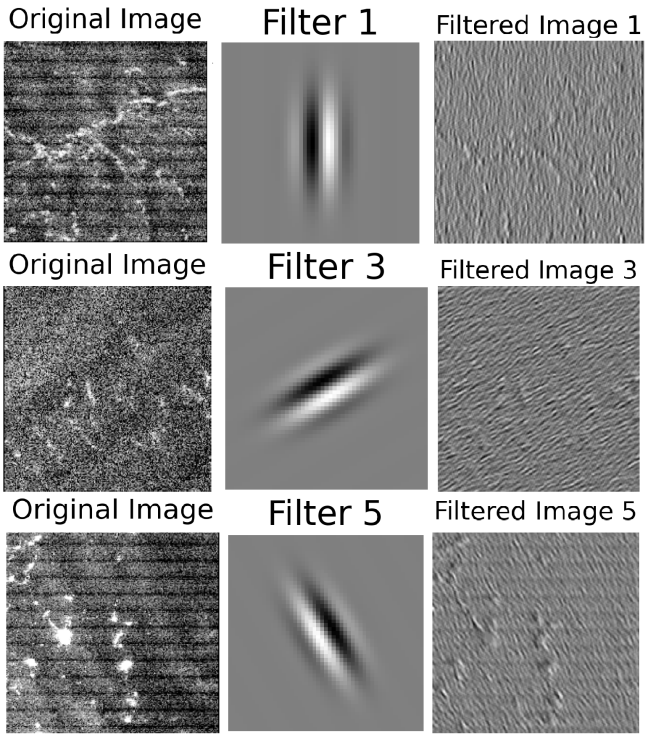


Fig: output comparison with input images using Gabor filter

For FFT, we also found inconsistency in terms of losing the gravity wave pattern instead of making them significant. In the following sequence images we compared two images (1, 3) with FFT converted patches (2, 4) where it seems to lose the actual patterns of gravity waves.

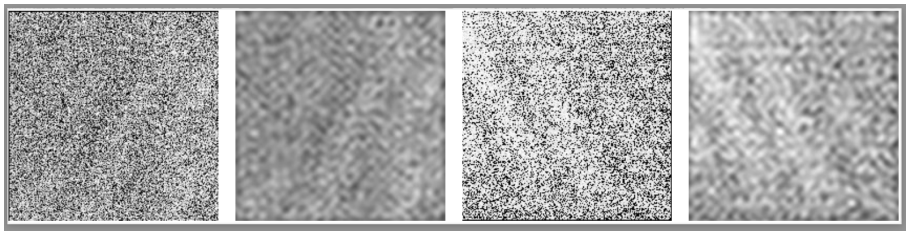


Fig: FFT converted images (image 2 and 4) with original images (1 and 3).

Model Configuration:
We further describe the ViT model as we customized from the base model. We used a custom ViT model as the base model did not learn anything. Following are the differences. Details are in /all source code/vit_no_custom_kernel_layer.py

the base model

```
patch_size=16,
num_heads=12,
num_layers=12,
image_size=224,
mlp_dim=3072,
```

the custom model:

num_classes = 2
patch_size = 40
num_heads=6 # Number of attention heads
num_layer=6 #Number of attention layer
image_size=200
mlp_dim=1024
att_size=32
batch_size=128
epochs=2000

We also show the model execution time using single GPU

model_name	execution_time (approx.)	node_type
ViT	22 hrs	single GPU
Gabor	24 hrs	single GPU
Sobel (trainbele)	30 hrs	single GPU
Sobel (non-trainbele)	28 hrs	single GPU
Laplacian (trainbele)	37 hrs	single GPU
Laplacian (non-trainbele)	34 hrs	single GPU
FFT (all but trainable)	26 hrs	single GPU
FFT Trainable	32 hrs	single GPU
gWaveNet (prior-kernel-applied)	39 hrs	single GPU
gWaveNet (no-trainable-layer)	42 hrs	single GPU
gWaveNet (with-traible-layer)	45 hrs	single GPU

Please find the source code files in the “source code directory”.