Meteorological Super-Resolution vs Wind Representations

Gruppo 21
Marzia De Maina, Matteo Galiazzo, Federica Santisi

Alma Mater Studiorum - Università di Bologna

Dipartimento di Informatica - Scienza e Ingegneria (DISI)

Prof. **Fabio Merizzi**

Purpose

The purpose of the project is to study the impact of different wind representations in the context of super-resolution.

Problem explanation and theory recap

[1]

Datasets and Preprocessing

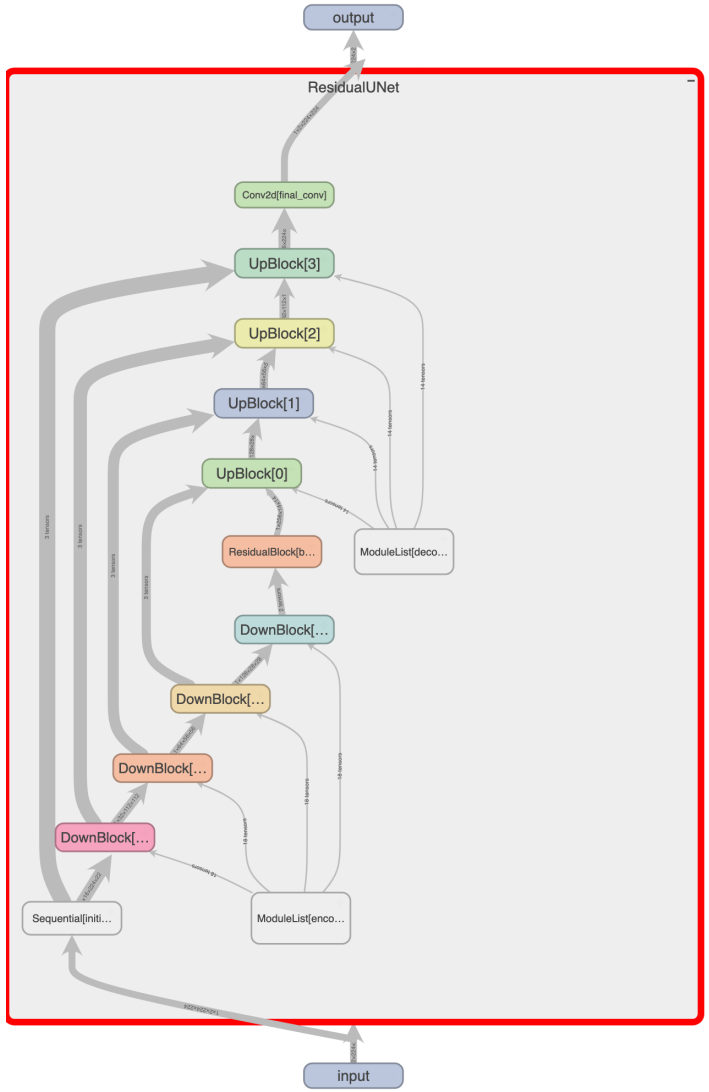
Model and hyperparameters

Our model is based on the U-Net architecture, extended with residual connections for better gradient flow and stability. The input has 2 channels (u, v) and the output also has the same 2 channels representing the high-resolution version of the same fields. The key building blocks are:

- **Residual Block**: it's the core module of the architecture. It has two paths
  - One that applies convolutions and normalization.
  - One that does nothing (the residual application).

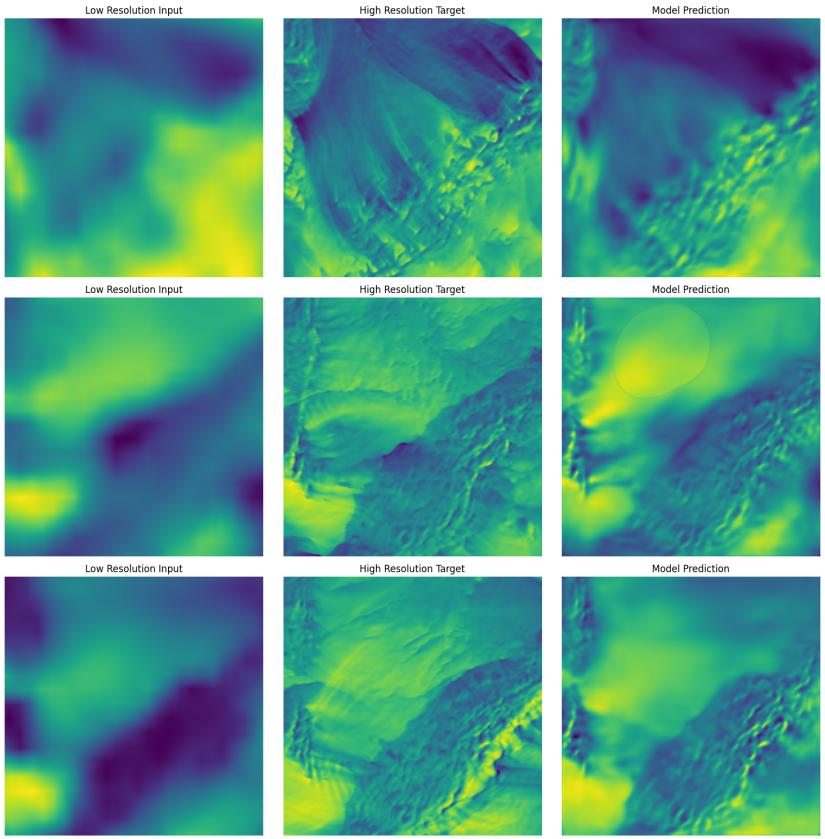
  The model learns how much to use either path.
- **Downsampling Block**: reduces the spatial resolution and increases the feature size. It uses stride 2 to downsample and includes a residual block.
- **Upsampling Block**: these are the mirror of down blocks. They use bilinear upsampling to increase the spatial dimension instead of transposed convolutions. Then, we concatenate the skip connections from the encoder and use a 1×1 convolution to reduce the number of channels after concatenations.

Model and hyperparameters

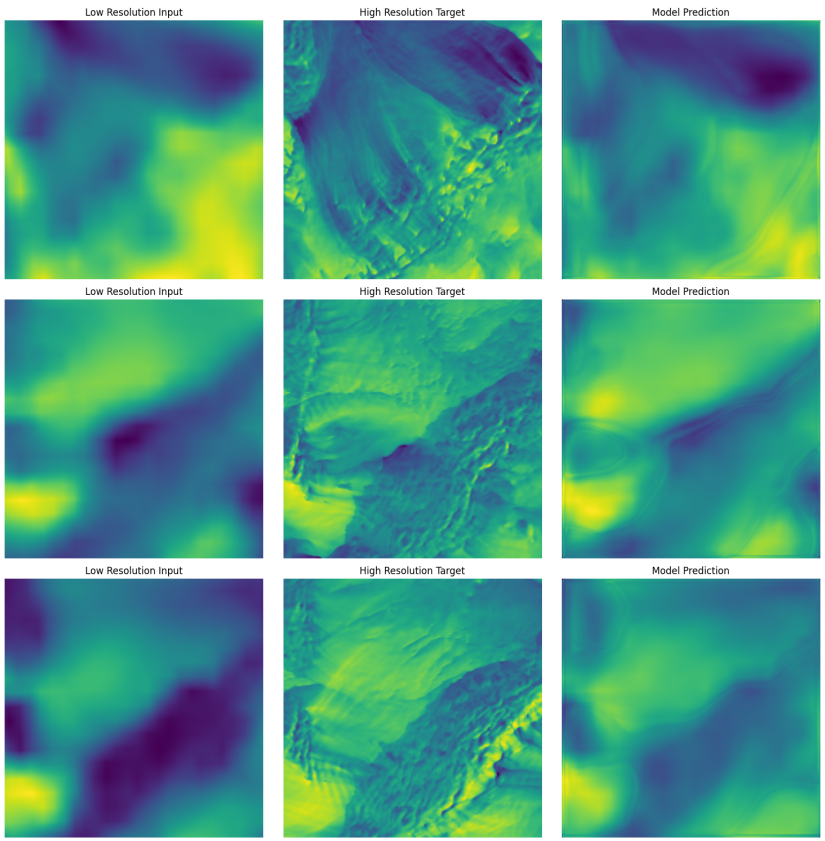


Model and hyperparameters

Even the best model couldn't get decent results with the suggested MSE loss, so we adopted a combined loss approach



L1 + SSIM loss. SSIM: 0.7556



MSE loss. SSIM: 0.7022

Model and hyperparameters

```
class L1SSIMLoss(nn.Module):
def __init__(self, alpha=0.85, ssim_window_size=11, ssim_data_range=1.0,
     ssim_channel=1):
    super(L1SSIMLoss, self).__init__()
    self.alpha = alpha
    self.l1_loss = nn.L1Loss() # Mean Absolute Error
    self.ssim_loss_fn = SSIMLoss(window_size=ssim_window_size, data_range=
         ssim_data_range, channel=ssim_channel)

def forward(self, y_pred, y_true):
    ssim_val_loss = self.ssim_loss_fn(y_pred, y_true)
    l1_val_loss = self.l1_loss(y_pred, y_true)

    combined_loss = self.alpha * ssim_val_loss + (1 - self.alpha) *
         l1_val_loss
    return combined_loss
```

Training

Results

Final Considerations

References

[1] Fabio Merizzi, Andrea Asperti e Stefano Colamonaco. "Wind speed super-resolution and validation: from ERA5 to CERRA via diffusion models". In: *Neural Computing and Applications* 36.34 (2024), pp. 21899–21921.