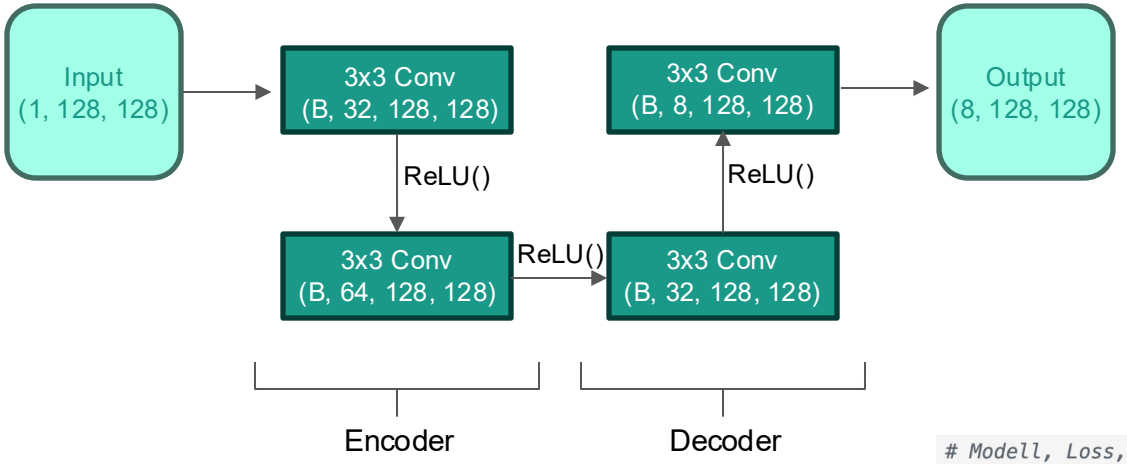
Solar Orbiter Images Challenge

Group: Machine Learning
Abhishek Anil Deshmukh, Jonas Hackfeld, Lu Meng, Paula Raith, Holger Stiele

Organization Among the Team

- **Strategy Alignment**: Agreement on using CNNs for the task
- **Dataset Preparation**: One member preprocessed and shared the dataset for consistency
- **Independent Model Development**: Each member experimented with different CNN architectures and hyperparameters
- **Result Comparison**: Models were compared, and the best were selected for refinement
- **Collaborative Optimization**: The team refined the top-performing model together

Ansatz 1



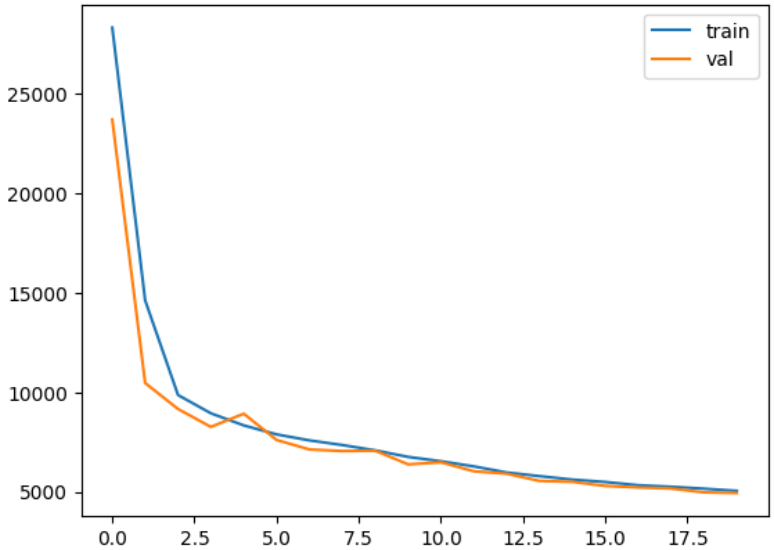

```python
class ChannelPredictorCNN(nn.Module):
    def __init__(self):
        super(ChannelPredictorCNN, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),  # (B, 32, 128, 128)
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # (B, 64, 128, 128)
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Conv2d(64, 32, kernel_size=3, padding=1),  # (B, 32, 128, 128)
            nn.ReLU(),
            nn.Conv2d(32, 8, kernel_size=3, padding=1)    # (B, 8, 128, 128)
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

```python
# Modell, Loss, Optimizer
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = ChannelPredictorCNN().to(device)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

Ansatz 2

- **Splitting Image**: Divide each image into four quadrants
- **Rotation**: Rotate each quadrant for variation
- → Reduces image size, making processing easier
- → Increases dataset size for more training data

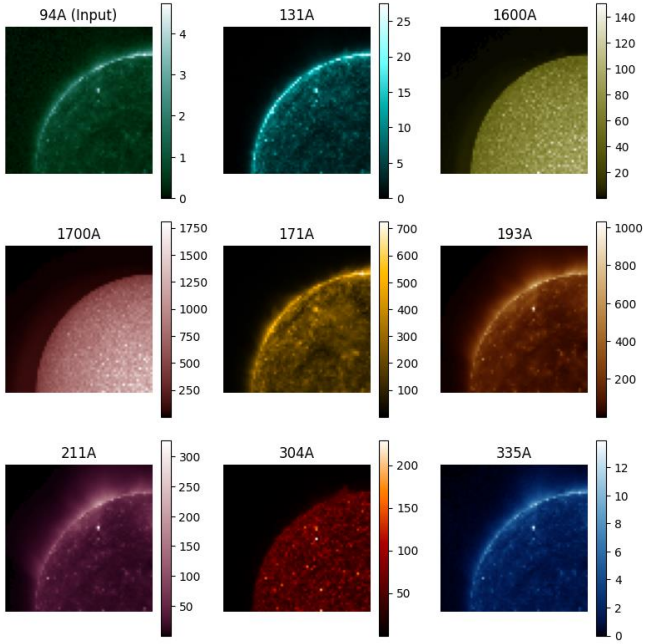
Results



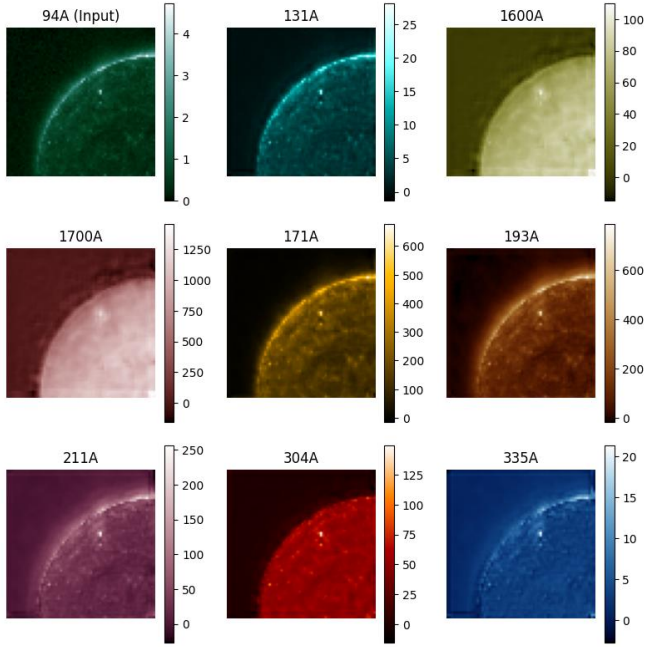
Comparison of Train and Validation Loss

ERUM DATA HUB

Results



Ground Truth



Prediction

ERUM DATA HUB

Environmental Impact

- Initially, our training process was not focused on consumption
- Once the architecture was figured out, we shifted our focus towards optimizing the training process and aimed for a balance between minimum training and maximum performance
- The goal was to reduce energy consumption while maintaining optimal performance

For training Quadrant Model: 0.03 kWh

For training whole images: 0.03 kWh

With minimal training quadrant model: 0.01 kWh

Outlook

- Instead of cropping the image into 4 parts we tried to use the transforms function in Torchvision.Transforms → didn‘t work yet
- Try different Architectures (e.g. add Residual block)