1 Developing a U-Net

The model developed for the two day prediction is based on the SimpleUNET architecture, though with a different sized Input layer to accommodate for the changed dataloader. The dataloader has subsequently been changed to appropriately select the correct fields from the .hdf5 samples and appoint them as input or target variables. As a result of using three variables of two days mean AA forecast, as well as sst, land-sea-mask and current time-step ice chart, the total number of predictors fed into the model is 9. Moreover, the resolution of all fields are kept at 1km, though their spatial extent is limited to (1920 x 1840). This resolution and spatial size conserves (almost) the entirety of the west-east axis of the AA domain. However, the southern border is raised by 450km compared to the AA domain. There are two main motivations behind readjusting the spatial extent of the predictors and targets.

Nevne størrelsen på grid, hva som blir sett på og hvorfor dette ble valgt som study area (operational coverage, U-Net downsample concerns)

1. The spatial extent of the input domain has to be divisible by the reducing factor enforced by the MaxPooling operation performed in the encoding component of the UNET.
2. The southern latitudes covered by AA has a proportionally skewed Sea Ice / Ice Free open water ratio, as exemplified in Figure (1). Increasing the southern bounding latitude of the subdomain thus decreases the number of guaranteed ice free pixels, which in turn decreases the skewness towards the ice free open water class for the UNET.

2 Model Architecture

The model architecture follows an encoder - decoder structure, commonly referred to as a U-NET [8] due to its shape funnelling the spatial data to coarser resolution, which resembles the letter ”U”. The current U-NET implementation follows that of Ronneberger et.al, though it has been modified with batch normalization after each convolution operation to ensure a more stable gradient flow. The weights of the model are Kaiming-He initialized [3], as the activation function used throughout the network is the ReLU function [6]. The final output of the model is a (1920, 1840, 7) tensor containing softmaxed probabilities along its final axis.

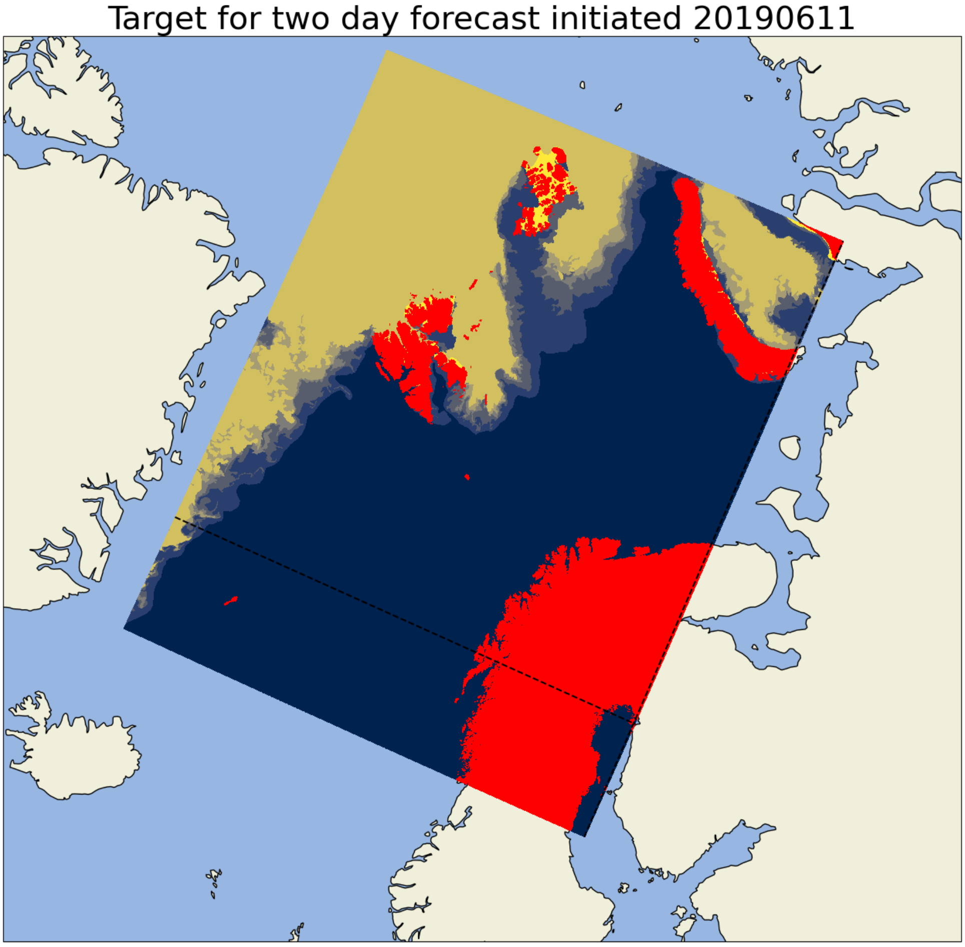


Figure 1: Example sample displaying an Ice Chart on a 1km Arome Arctic projection. Note the horizontal and vertical dashed black line which indicate the domain subsection used by the UNET

2.1 CategoricalCrossEntropy-Loss

As the title suggests, these runs of the model involved using CategoricalCrossEntropy as the loss function for multi-class image segmentation. Categorical Cross Entropy loss is defined as

$$CE = -\sum_{i}^{C} y_i \, log \left(\hat{y_i} \right) \tag{1}$$

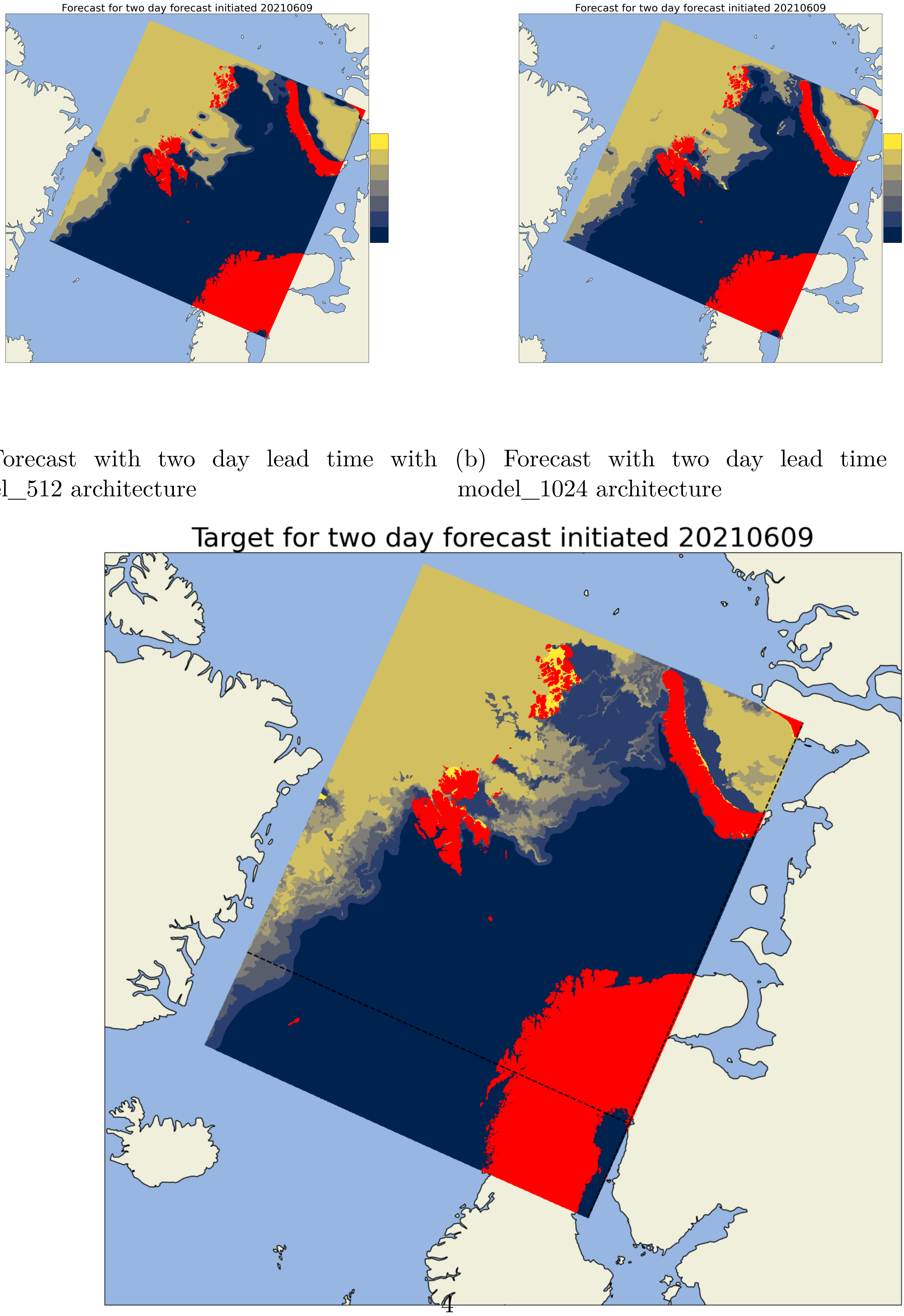
where C denotes the number of available classes, y the ground truth and $\hat{y}$ a prediction of y. Note that as y is onehot-encoded, the formulated function only contributes to the overall loss with the log of the predicted probability of the correct class according to the ground truth.

Two variants of the previously described model have been trained with the CategoricalCrossEntropy described in equation (1). The first model was trained with an encoder consisting of 4 convolutional blocks with channel dimensions (64, 128, 256, 512). The second model consisted of 5 convolutional blocks, with an identical architecture except for the last convolutional block increasing the channel dimension to (1024). Example outputs as well as target can be seen in Figure (2).

By inspecting Figure (2), two observations can be made. The first observation is regarding how the model complexity affects how it fit to the data. By comparing Figure (2a) with (2b), it can be seen that the latter is resolving the finer structures of the ice edge to larger extent than the prior. Though the overall correctness is left to be discovered, this shows that increasing the depth of the encoder (increasing the trainable parameter count from 7 million to 31 million) is reflected by the model preserving the details of the ice edge structure. Though it is non-trivial to say why the 1024-model preserves the details to a larger extent than the 512-model, it does follow from the U-Net architecture that a deeper encoder (higher channel count and more convolutional blocks) is better at describing "WHAT" is in the image compared to the shallow-layers, which include a larger amount of spatial information and tells the model to a larger extent "WHERE" things are in the model.

This may have a source

The second observation made from inspecting both forecasts is their inability to represent classes 2 and 3. This likely arises from the general movement-pattern of the sea ice, where the intermediate classes are much less likely to appear than the edge-most classes. Furthermore, the sea ice is much more likely to represent a wider range of concentration classes in the intermediate ice edge region over time, making it more difficult for the network to confidently predict those classes compared to the more probable classes. As can be seen by the network immediately predicting class 4 after class 1, creating an


Forecast for two day forecast initiated 20210609
Forecast for two day forecast initiated 20210609
Target for two day forecast initiated 20210609

(a) Forecast with two day lead time with model_512 architecture

(b) Forecast with two day lead time with model_1024 architecture

(c) Target for forecast with two day lead time

Figure 2: Example forecast attempt made by model_512 and model_1024 09-06-2021

artificial cut-off region. However, to what extent the intermediate classes are predicted has not been inspected directly , though it is likely to assume that they are predicted though with a lower confidence than that of class 4 (which is consequently why it is visualized, as the most probable class is chosen regardless).

They should be

2.2 FocalLoss

The focal loss is derived as a generalization of the Cross Entropy Loss listed in Equation (1). The intent of the loss function is to downweight the easy to predict samples, while focusing on the hard to predict samples by allowing their gradient to have a higher impact on the network [4]. Mathematically, focal loss is defined as

$$\mathrm{FL} = -\sum_{i}^{C} \alpha_i (1-\hat{y}_i)^{\gamma} y_i \, log\left(\hat{y}_i\right) \tag{2}$$

where α is a balancing parameter, γ is the focusing parameter ($\gamma = 0 \rightarrow CE$), with the rest similar as Equation (1).

Include figure showing focal loss output, discuss implications of using this loss function

By inspecting Equation (2), it can be seen that predictions that the model is quite confident in making, i.e. $\hat{y}_i \rightarrow 1$ send the Focal Loss towards zero. For the current application, the assumptive motivation is that this affects (by reducing) the contribution made by the Ice Free Open Water pixels as well as the Very Close Drift Ice (class 6), which are the most represented classes in the CE loss model seen in Figure (2). Consequently, as the loss contributions of the most likely (and most represented classes) is reduced, the harder to predict (both due to being less represented and due to sea ice movement) have a larger impact on the overall loss propagating backwards throughout the model. As a result, these intermediate classes should be predicted as the most likely class, resulting in a less sharp ice edge which closer represent the Ice Charts.

2.3 Cumulative probability distribution model

Discuss difference in dataloader, same dataset is used differently

2.3.1 Separate convolutionial layers as output

Data exists, start writing

2.4 Model Selection

During the training of a deep learning system, there exists several different ways to save a state of the model during training. A naive approach would be to let the model train all predetermined epochs, and save the weights of the model at the end of the final epoch. However, this approach would be indifferent to whether the model has converged, generalized or overfitted and is thus an inadequate way to save the weights. The Tensorflow Keras API supplies functions which can be used customize the training loop in the form of callbacks, with the EarlyStopping and ModelCheckpoint callbacks relevant for model selection [5]. EarlyStopping is a technique which ends the training loop when it detects that a monitored values has stopped decreasing. On the other hand, ModelCheckpoint continuously saves the model if a certain condition is met, without terminating the training loop. Both callbacks support monitoring the validation loss as the metric in which to optimize the model. However, a custom metric such as yearly mean IIEE [2] could be monitored instead.

To aid in model selection, I developed a custom callback which computed the Normalized IIEE with respect to a climatological Ice Edge length derived from ten years of OsiSaf data , following the observation in 2 that IIEE is correlated across spatial resolutions. The callback computes said metric for all samples and reduces them to a yearly mean of the validation set. Similar to the aforementioned callbacks, the developed callback is executed at the end of an epoch where it computes the mean Normalized IIEE for all predicted samples from the validation set, which it appends to the *logs* dictionary used by Tensorflow to keep track of other computed metrics, such as loss and validation_loss for the current case. Thus, the newly developed callback would allow for model selection based on Normalized IIEE, as well as the already computed validation loss.

Write about the climatological Ice Edge dataset, ref section from here

When comparing different models to asses their performance, this project will frequently compare their Normalized IIEE as the metric is Normalized by the ice edge, thus reducing the seasonal variability of the Metric [7] . As such, it would be beneficial to select a model based on its Normalized IIEE validation performance. With the above callback, such a selection is possible. However, including the IIEE verification metric as is done in the above callback increases training-time of ten epochs from ≈ two hours without the IIEE callback to ≈ 24 hours with the IIEE callback. As 20 epochs is currently an adequate number of epochs at the time of writing , it would be too computationally costly to select a model based in its validation Normalized IIEE performance.

This citation is actually for SPS$_{\text{length}}$ but SPS is reduced to IIEE for a deterministic ice edge [1]

On the other hand, it can be seen by inspecting Figure (3) that the Normalized IIEE tend to evolve conjunctionally with the validation loss, in the current case defined as the mean cross entropy of all validation samples. Furthermore, the validation loss and Normalized IIEE in Figure (3) have a correlation of 0.82 with regards to epoch. Note that this has been calculated only using the numbers present in Figure (3). As such,

This

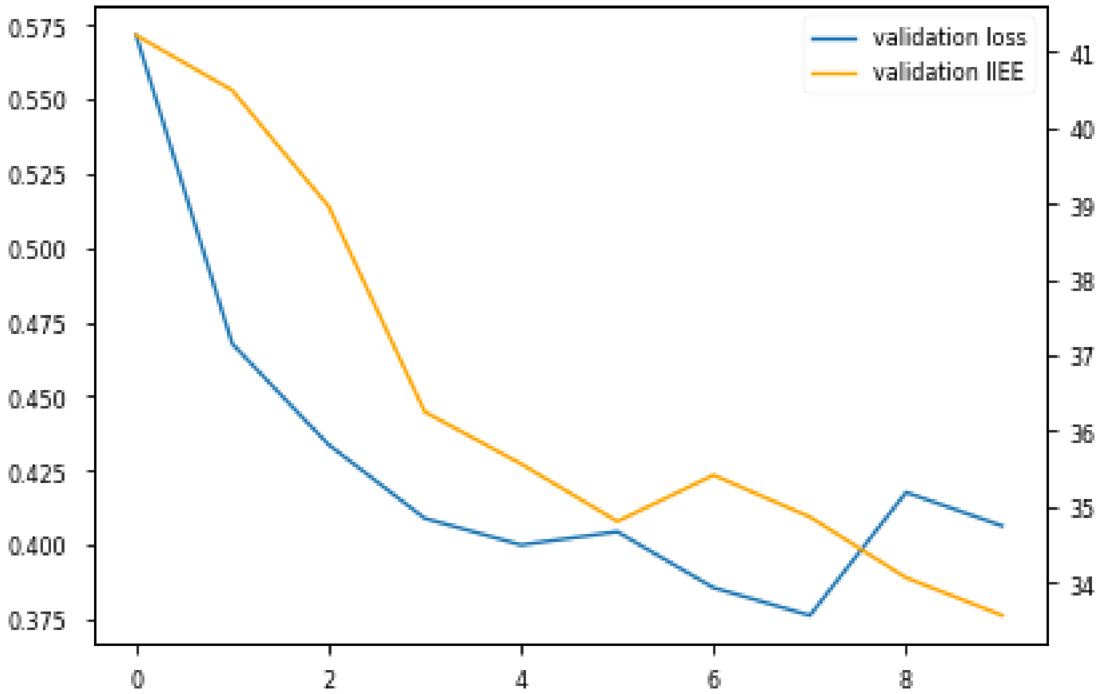


Figure 3: validation loss and Normalized IIEE computed as mean of validation set for each epoch during training

there is reason to believe that selecting a model based on its validation loss, which is quick to compute, would result in a generalized model which may also excel at lowering its Normalized IIEE.

When selecting the best model, this project will apply the ModelCheckpoint callback with regards to validation loss as outlined above. ModelCheckpoint is preferred compared to EarlyStopping, as interrupting the training loop early may result in an ”undercooked” model. E.g. the weights in earlier model layers are adjusting slower than later weights, giving the impression that the model training has reached a plateau which causes the model to stop. Whereas if the model where to continue training, the later adjustment of earlier weights would cause a later spur in increased model performance. ModelCheckpoint was chosen since behavior such as what was just exemplified is possible with the callback.

References

[1] H. F. Goessling and T. Jung. “A probabilistic verification score for contours: Methodology and application to Arctic ice-edge forecasts”. In: *Quarterly Journal of the Royal Meteorological Society* 144.712 (Apr. 2018), pp. 735–743. DOI: 10.1002/qj.3242.

[2] H. F. Goessling et al. “Predictability of the Arctic sea ice edge”. In: *Geophysical Research Letters* 43.4 (Feb. 2016), pp. 1642–1650. DOI: 10.1002/2015gl067232.

[3] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. DOI: 10.48550/ARXIV.1502.01852.

[4] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: (Aug. 2017). arXiv: 1708.02002 [cs.CV].

[5] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[6] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *ICML*. 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[7] Cyril Palerme, Malte Müller, and Arne Melsom. “An Intercomparison of Verification Scores for Evaluating the Sea Ice Edge Position in Seasonal Forecasts”. In: *Geophysical Research Letters* 46.9 (May 2019), pp. 4757–4763. DOI: 10.1029/2019gl082482.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: (May 2015). arXiv: 1505.04597 [cs.CV].