A Deep Learning approach to detecting cloudiness levels in all-sky images

Student: Tom Greenwood (tomgreenwood10@gmail.com)

Summary

A deep learning approach has been taken to segment all sky images and identify the clouds. A u-net convolutional neural network was built and can segment the images with an accuracy of 0.89.

The problem

The British Antarctic Survey have all-sky visible light cameras which operate at 2 Antarctic sites and in Cambridge. The cameras take images at frequent intervals. The aim is to use deep learning to determine the cloudiness levels in the images. This could be used to allow the automatic sorting of cloud sensitive datasets. The 2 specific objectives are:

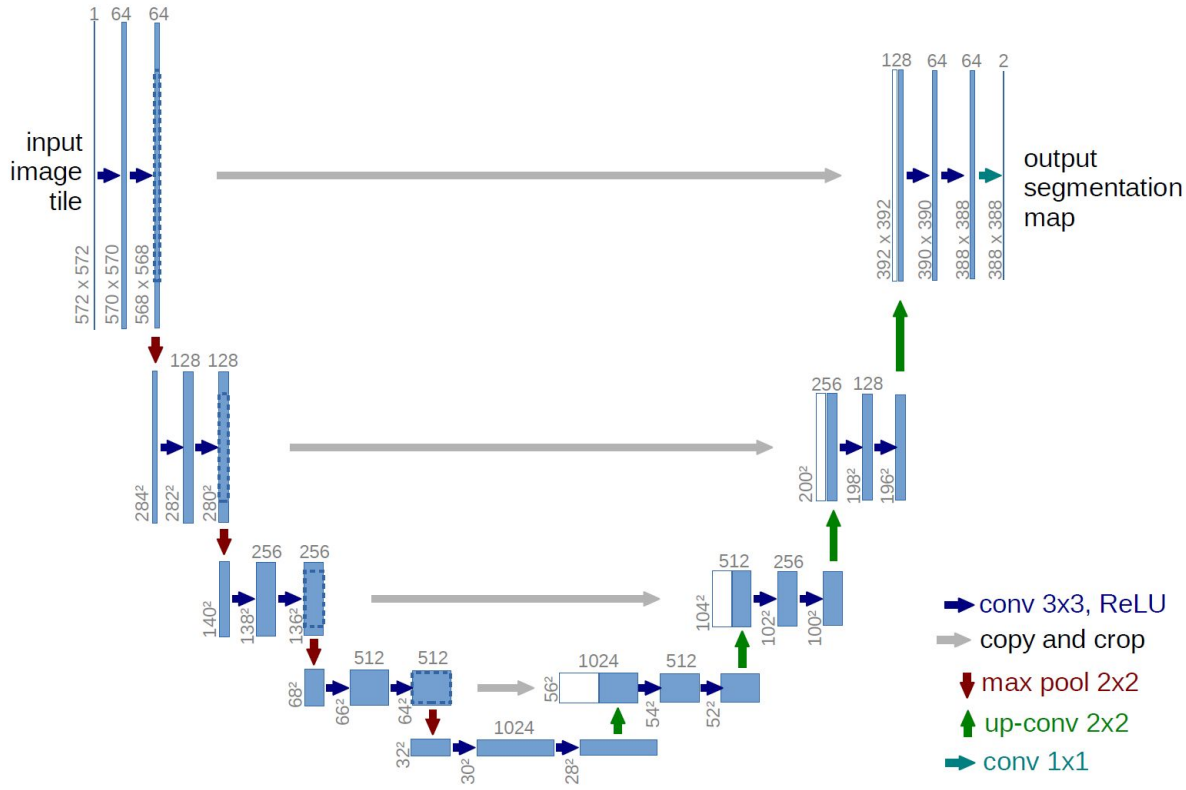
- Determine overall cloudiness level per image.
- Determine if there is cloud cover at the zenith per image.

Data modelling and approaches taken

In order to solve both objectives, semantic segmentation will be required. In addition, this may represent a more complete image understanding than a classification approach, which could lead to further improvements in later versions of the model.

As the aim of this project is to explore a deep-learning approach a convolutional neural network (CNN) will be used. Neural networks can have many different architectures. For image segmentation; one requirement is that the network predicts an image, pixel by pixel, the same size as the original / input image. The U-Net architecture proposed by Ronneberger et al in 2015 does this and has shown very good performance on segmentation of biomedical images, winning the ISBI challenge in 2015.

The network architecture comprises of an encoder and decoder. The encoder closely resembles CNN structures used for image classification e.g. VGG16. This applies blocks of convolutional layers with (3, 3) filters followed by pooling layers to reduce the size. After each pooling layer the number of filters in the subsequent convolutional layers is doubled. This happens several times until the output shape is considerably reduced from the starting shape. The decoder then rebuilds the size by upsampling and further convolutional layers, in order to get a prediction on each pixel and is nearly a mirror image of the encoder. It is important however to give the decoder access to the earlier stages of the encoder, where fine details of the image where captured. This is achieved by concatenating output layers in the decoder with the output layer in the encoder of the corresponding size before feeding this into the next convolutional layer. A schematic of the Ronneberger U-net structure is shown below:



The network used in this project has the input sizes adjusted and also the number of filters reduced to overcome memory problems encountered with the hardware available.

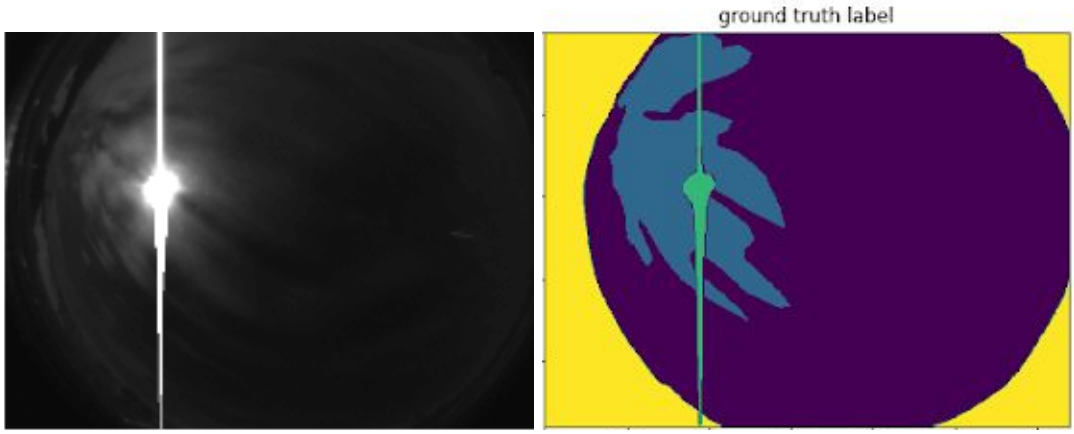
All model training was carried out on a google cloud virtual machine with 1 Nvida Tesla P4 gpu. Keras with a Tensorflow 1.14 backend was used.

To train the model 84 images with corresponding labels (annotated images) were used, an additional 7 were used for validation (0.1 validation split) and 7 images were reserved for testing.

Challenges and assumptions

The labelling process is critical to the model performance and in this case was quite difficult. It was unclear in many images exactly what was cloud and what wasn't - therefore there will be inaccuracy built into the labelling from the start. It was difficult to identify the edge of high, thin clouds and also cloud edges in relatively clear night skies. In addition, I am not a domain expert so some miss-identification is likely.

An example of a difficult to label image is shown below where different labellers may label the cloud edges in alternative locations (*note; the image brightness has been increased for clarity.*)



I have generally taken the strategy of labelling pixels as cloud when I am reasonably sure there is cloud there, however this perception can differ depending on the image - for images dominated by cloud it is often easier to identify the clear sky and subsequently the cloud via subtraction. In addition the adjusted brightness of the image being labeled can also have an effect. It may be beneficial to standardize the labelling procedure in order to improve the model further. For reference I used Labelbox for labeling, and adjusted the images to maximum brightness.

Many images contained an area of pixels saturated at maximum brightness by the sun. As it is unclear if there is cloud or clear sky in these locations therefore 'sun' was used as a label category in order to exclude these pixels from final percentage coverage calculations.

The categories used for labelling were:

Category	**Pixel value on labelled image**
clear sky	0
cloud	1
sun	2
surround / background	3

Using the segmentation template on Labelbox clear sky category was not labeled, which left these pixels as value 0.

It may have been possible to label the surround / background with a rule based function as this was in the same location on all of the sample images I inspected. I have, however, labeled this

alongside the other categories as a model which can identify the surround intelligently may be more transferable to other cameras.

Base model build

Inputs images into the model were preprocessed with the preprocess_labeled_data function in the model module. This normalises the images (either by the whole array or image by image - user defined) combines them into an array and reshapes this for input into the model. The following image augmentations are available in this function:

- Vertical flip
- Horizontal flip
- Brighten and darken
- Gausian blur

If augmentations are chosen, all images are copied and the augmentation applied to the copy giving every combination of augmentations in the final preprocessed array. Augmentation can increase the training size and also make the model more robust to changes in the image which may not affect human ability to identify categories. For all model training in this project I have used horizontal and vertical flips as augmentation but not brightness or blur augmentations. The flipping creates a similar image but with the categories in different places, therefore increasing the training size and hopefully making the model more robust to positional changes in the categories (which may be seen if the camera is moved / from a different location). The brightness augmentation was not used as this would have been affected by or cancelled out by the image normalisation. The blur was also not used for time purposes however these options have been left in the function in case of further experimentation.

Experiments

The general model structure was developed iteratively until reasonable results were obtained. At this point 3 experiments were run with the aim of optimising hyperparameters or the data pre-processing method. These factors were investigated one at a time. Due to the unstable nature of model training (the same model may perform very differently in training due to different parameter initialization) and the time available, it was not possible to identify interactions between these factors which would have required a larger experiment size. Identification of interaction effects between factors could be performed as further work to further optimise the model.

Each experiment was analysed in a similar way. The evaluation metrics used were accuracy and loss. Visual examination of predictions on the test data were performed on the final model.

Note: A helper function utils.training_bands() was used to plot the training metrics. This shows the metrics for multiple iterations of the same model and fills the area between the highest and lowest data.

Experiment 1 - Learning rate

The learning rate is a scalar to the step size taken with each optimisation. A larger learning rate will result in larger steps and often a faster convergence of the model (less epochs). This however is not always the case as large steps can overshoot minima[1] and small steps can get stuck in local minima. Therefore this first experiment will fit models with 3 learning rate values and assess which appears the most suitable.

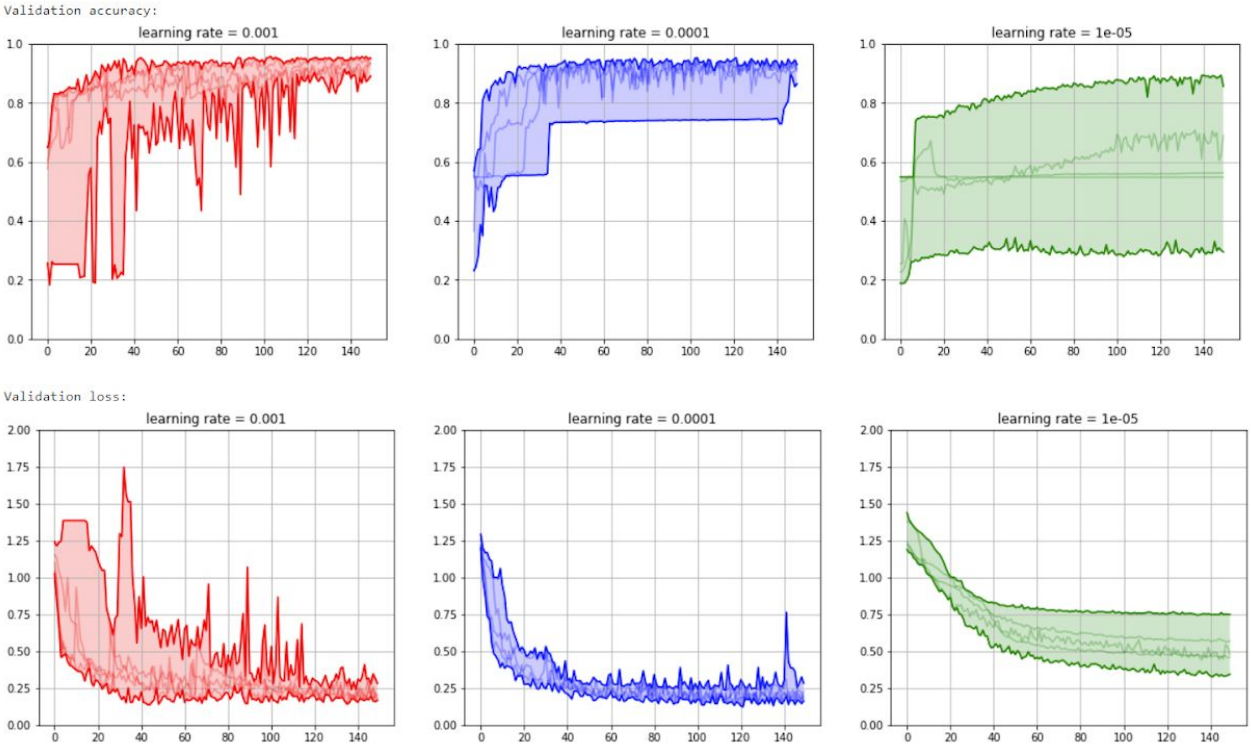
Models sometimes benefit from reducing the learning rate once plateaued. This may be done in the later stages of prototyping - this experiment will just look for the best starting rate.

Although fast model training may be advantageous for some applications, it does not have a major impact here. Overall model accuracy / good performance on metrics is the aim, as well as avoiding plateauing or in the worst case no model convergence at all.

Due to initial parameter randomisation, training models with the same parameters can often result in different training patterns. Therefore 5 training iterations will be performed with each learning rate.

Evaluation metrics vs training epochs are shown for all learning rate and iterations below:

[1] ‘Minima’ is used throughout and refers to minima in the loss function.



Observations:

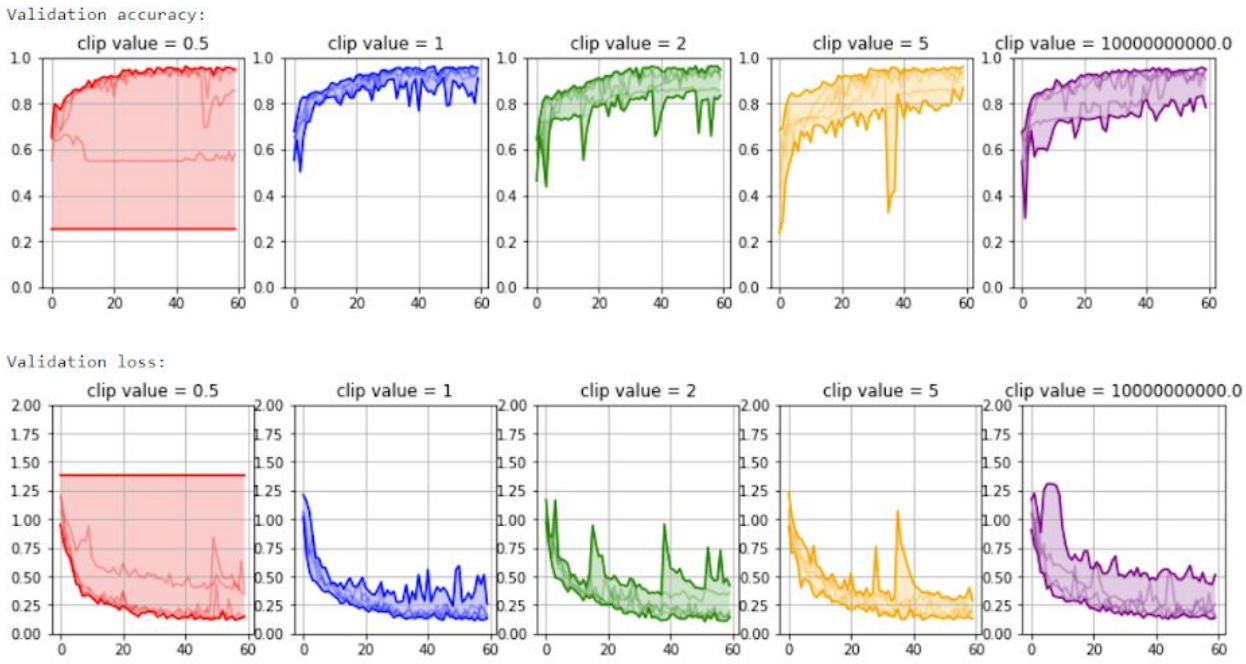
- The lower learning of 1e-5 rate shows possible plateauing on multiple iterations.
- The best iterations from learning rates 0.001 and 0.0001 achieve similar values by epoch 150.
- The largest learning rate of 0.001 has some iterations that show instability.

It looks like a learning rate of 0.0001 may be the most stable, however the best models from both 0.001 and 0.0001 show similar performance but the maximum performance is reached quicker with the higher learning rate - this is no surprise but what it proves is that the higher learning rate is unlikely to be overshooting loss minima, at least most of the time. I will progress with a learning rate of 0.001 to speed up / shorten some of the next experiments but the final model may have a starting learning rate of 0.001 which can be reduced when it plateaus.

Experiment 2 - Clip value

Gradient clipping is the act of limiting parameter gradients in the network. The clip value parameter ('clipvalue') sets a limit the magnitude of parameter gradients. It can be considered a form of regularisation. It can be useful if large gradients in the function are found as a non-limited parameter gradient will result in a large step, potentially overshooting a minima.

Models were training with clip values of 0.5, 1, 2, 5 and 1e10 (effectively no clip value). 5 iterations of each clip value were trained and the evaluation metrics are plotted against epoch below:



Observations:

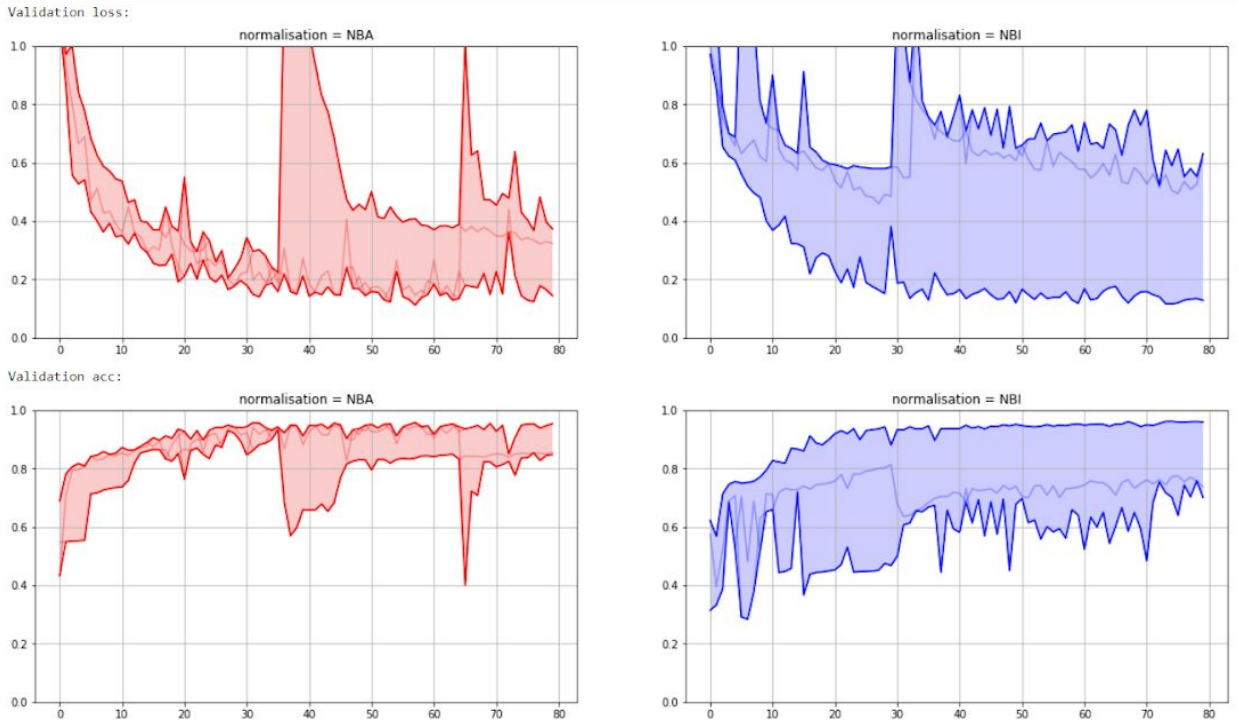
- The highest performing iterations for all clip values reach similar performance.
- Clip values 0.5 has iterations that show some signs of plateauing.
- Clip value of 1, 2 and 5 show similar performance.
- No clip value (set at 1e10 here) shows reduced performance on some iterations.

I will progress with a clip value of 5 in order to provide some regularisation but not too much restrictions in case there is interaction with other factors.

Experiment 3 - Normalisation strategy

So far, all the experiments have been run with the input array (X_train) normalised[2] over the whole array. This will keep the darker images darker and the lighter lighter. It's unclear if this is beneficial, so this experiment will compare this approach verses normalising each image individually, which will give each image the same average brightness.

The plots below show the evaluation metrics vs epoch for each normalisation strategy (NBA = normalised by array, NBI = normalised by image). 3 iterations of each condition were trained.



Observations:

- Both normalisation strategies whow instability in some models.
- Performance is similar in the best performing models of each.
- Normalised by image (NBI) shows less instability on it's best performing model.

There may not be a major difference here. I will progress with images normalised individually as this should make the model more robust to extremes in bright or dark images (despite not being evident here).

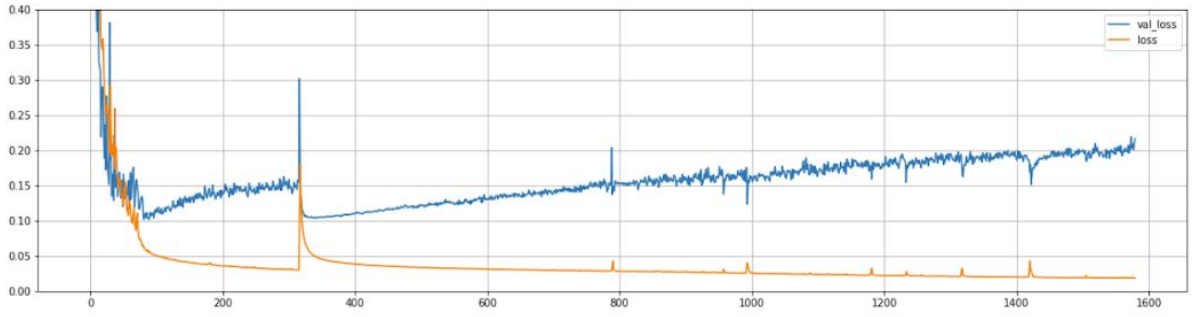
[2] Normalisation included subtracting the mean and dividing by standard deviation.

Final model

Training

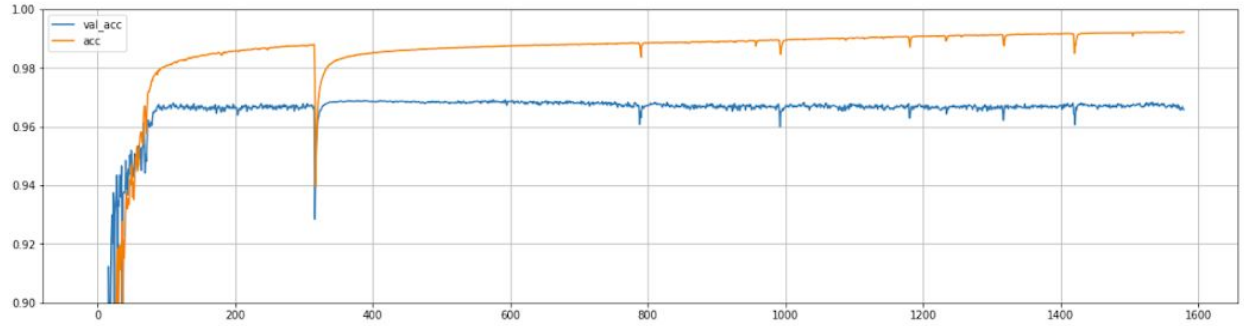
A final model was trained with the best hyperparameters and pre-processing strategies found in the experiments. The best model from experiment 3 (normalised by image, iteration 3) utilized these parameters therefore was taken and the training continued for 1500 additional epochs at a reduced learning rate or 0.0001, totalling 1580 epochs. All models have been set to save the best model, defined by having the lowest validation loss, from the training. Therefore worst models from later epochs will not overwrite earlier models if they are not as good and should ensure that the model saved is not overfitted.

The loss and validation loss from the training is shown below.



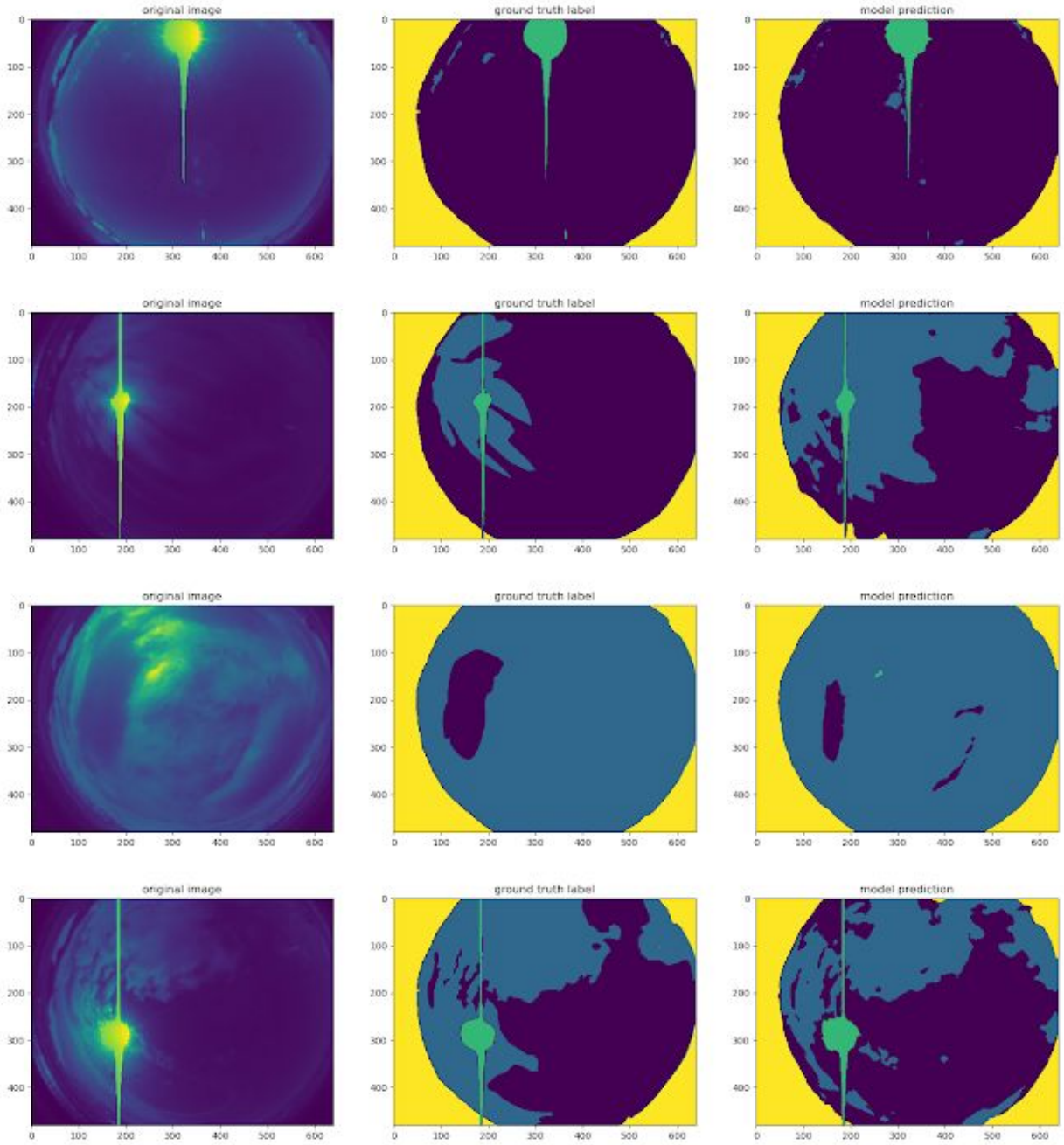
For much of the training a divergence between the loss and validation loss can be seen. This indicates overfitting from early epochs. Therefore with the current training data, models can reach maximum performance within 100 epochs. This may change if the training data size is increased.

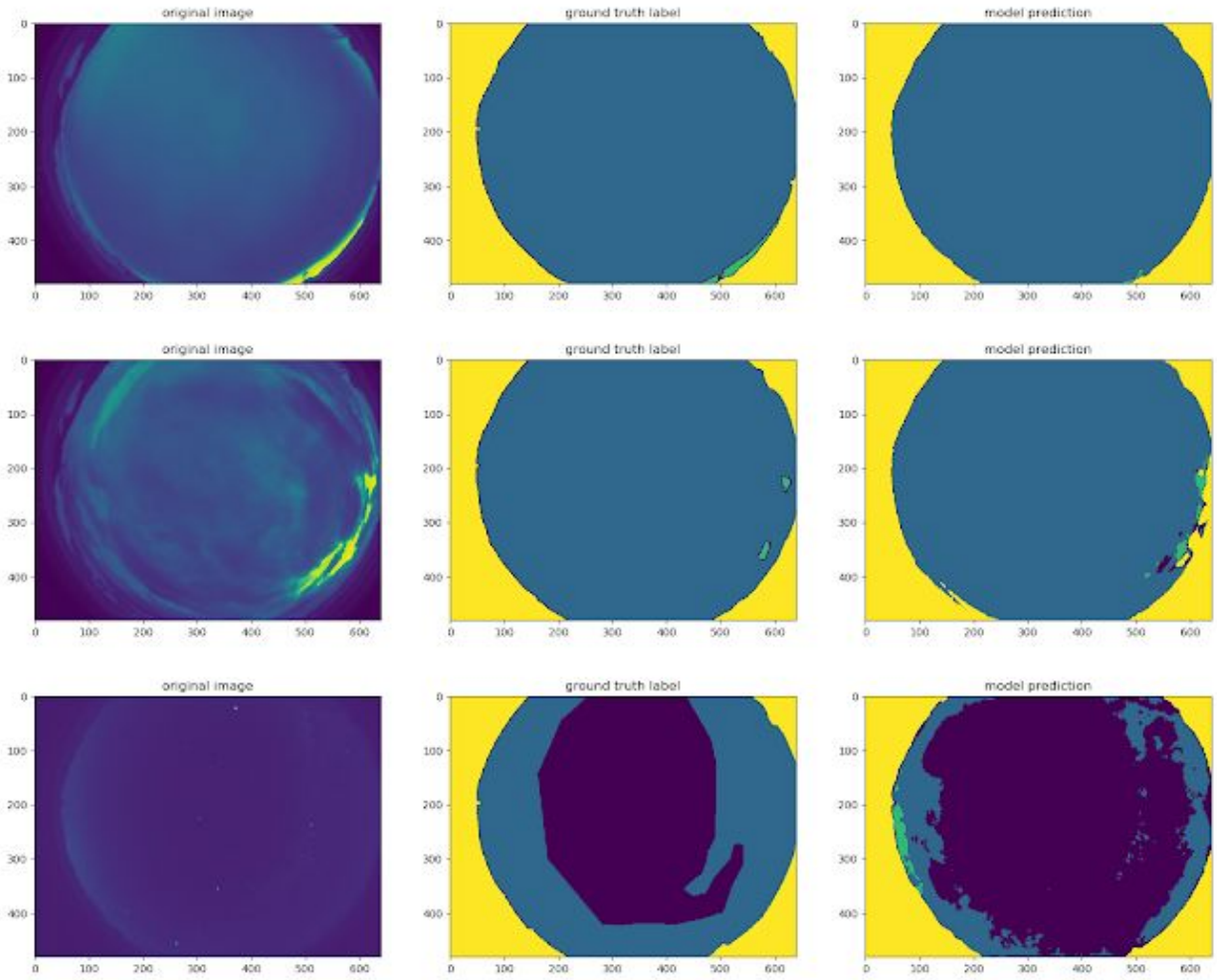
Similar behavior can be seen in the accuracy vs epoch shown below:



Results of the data analysis

Visual examination of the models predictions on the test images are shown below:





Reasonable similarity can be seen between the ground truth labels and the model prediction. The test image include mostly clear night sky, fully cloudy, high thin clouds, and mostly clear day sky.

Some misclassification of clear sky as cloud can bee seen on the first image. This will not greatly affect the percentage cloud cover calculation but could lead to misclassification when cloud cover at the zenith is predicted. If this specific misclassification occurs it may now be too disadvantageous as it will provide conservative estimates of cloud cover at the zenith where it is more likely to classify as cloudy than clear.

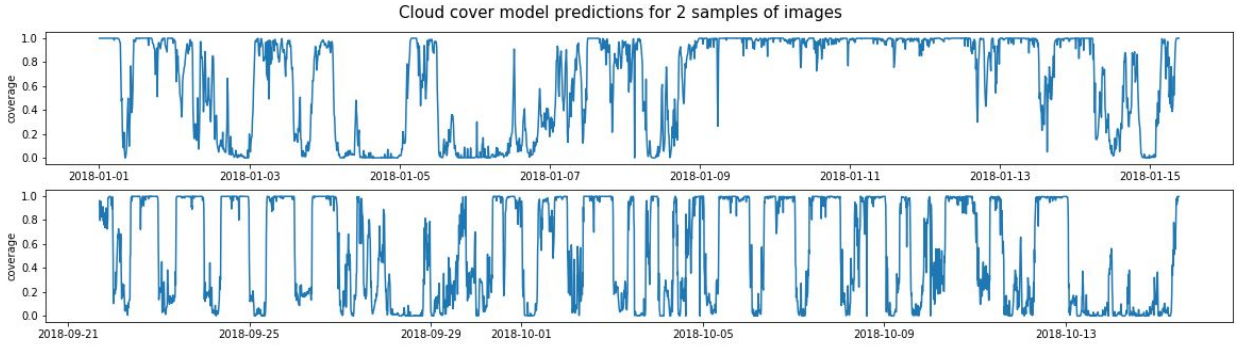
The second image also shows over prediction of the cloud category however this image is an example of thin high cloud and the final prediction looks accurate when compared to the input image.

A classification report (generated by sklearn.classification_report) for the cloud category vs all other categories on the test data is shown below:

	precision	recall	f1-score	support
not cloud	0.89	0.90	0.90	1160783
cloud	0.88	0.87	0.88	989617
accuracy			0.89	2150400
macro avg	0.89	0.89	0.89	2150400
weighted avg	0.89	0.89	0.89	2150400

This shows reasonable performance on all metrics and can be used to benchmark future model against.

Model predictions from the 2 samples of images (2 separate time periods) were made and are shown below.

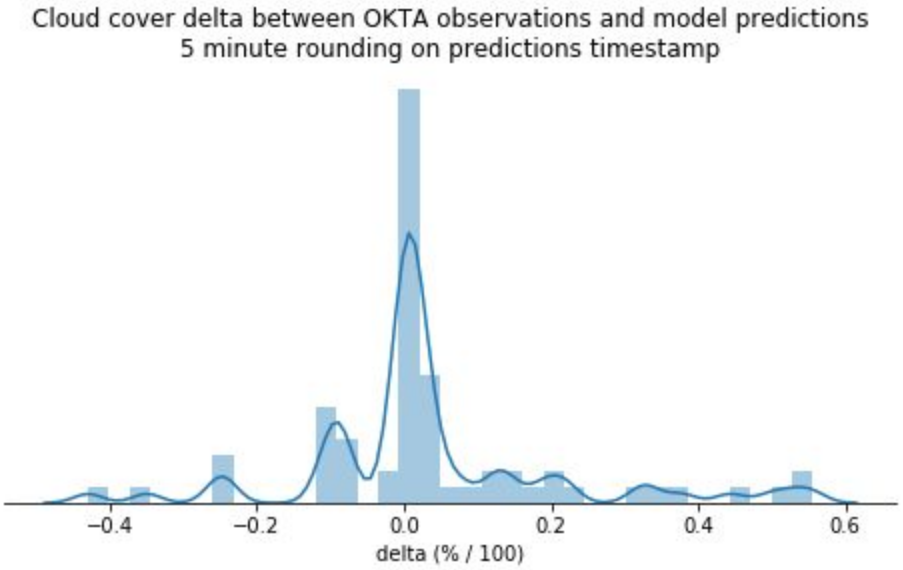


The predictions show some structure vs time and are not random. This may indicate accuracy in the predictions. To correctly filter out time periods from other instruments datasets, it may be possible to improve the accuracy by considering the images predictions as a moving window over time. A prediction that shows a clear sky may be more reliable if the surrounding predictions also show clear sky than if not.

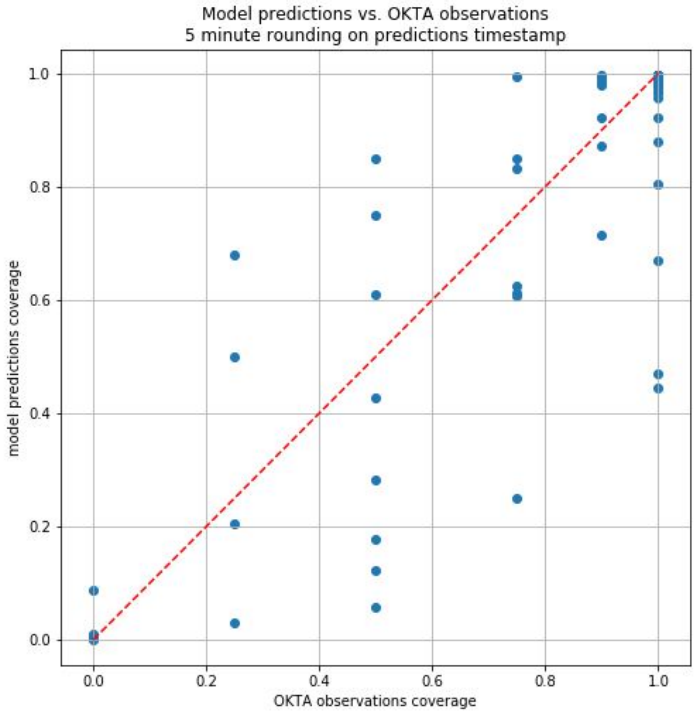
Comparison to additional data

Additional data was also available to benchmark the prediction against. The OKTA observations are 3 hourly visual observations by a human and record the cloud coverage in one on 8 categories ranging from no cover to full cover.

2 time periods of images were sampled and their timestamp rounded to the nearest 5 minute interval. This gave 68 images that correspond to a time where a OKTA visual observation was also taken. The difference / delta between the OKTA recording and the model prediction was calculated and the distribution of deltas is shown in the figure below.



This shows a concentration of deltas around zero with little skew in the distribution. This indicates reasonable performance however some delta range to over 0.5 (50%) different. The predictions vs the OKTA observations are shown in the scatterplot below.



This shows a weak relationship between the predictions (pearson correlation coefficient = 0.85) and the OKTA observations, therefore the model predictions should still be treated with care as some major inaccuracy is found in a minority of predictions.

Comparison to cloudbase laser detection

The cloud cover at the zenith, for predictions, has been calculated by taking the mean of the pixels in a 10 by 10 pixel patch in the centre of the image. With clear sky a 0 category and all other categories being a higher number, the mean prediction value of this patch should indicate the likelihood that it is clear sky. The mean values were then converted to a boolean with a threshold of 0.5, above which it was assigned a value of 1 indicating that cloud was present, below a value of 0 indicating clear sky. The cloudbase reading has multiple categories, with only category 0 indication no cloud detected. These were also converted to a boolean with all values above 0 being converted to 1 to indicate cloud was present.

Image timestamps were rounded to the nearest second which gave 177 instances where an image was taken at the same time as a cloudbase reading. In 82 % of instances the predictions agreed with the cloudbase readings.

Conclusions

- The model exhibits reasonable performance on the sample images.
- The model predictions correlate to the OKTA observations with a pearson correlation coefficient of 0.85.
- The model predictions agree with the cloudbase laser reading in 82 % in a sample of 177.

Recommendations for further work

The following suggestions for further work mak increase the model accuracy:

Increase training size -- Neural networks will generally show an increase in performance if the size of the training dataset is increased.

Improve image labelling -- If the accuracy of the image labelling is increased this should increase the accuracy of the model. Implementation of standard labelling procedure may be beneficial. In addition, alternative labelling tools should be trialed. The Labelbox tool used in this project assigned a value of 0 to some pixels at the intersection of 2 categories. It is likely that the model has tried to replicate this false behaviour.

Further hyperparameter and model architecture optimisation -- This may improve model performance. Candidates are; the optimiser used, the number of filters used in the convolutional layers, the number of convolutional blocks.