ABSTRACT: The identification of atmospheric rivers (ARs) is crucial for weather and climate predictions as they are often associated with severe storm systems and extreme precipitation, which can cause large impacts on society. This study presents a deep learning model, termed ARDetect, for image segmentation of ARs using ERA5 data from 1960 to 2020 with labels obtained from the TempestExtremes tracking algorithm. ARDetect is a convolutional neural network (CNN)-based U-Net model, with its structure having been optimized using automatic hyperparameter tuning. Inputs to ARDetect were selected to be the integrated water vapor transport (IVT) and total column water (TCW) fields, as well as the AR mask from TempestExtremes from the previous time step to the one being considered. ARDetect achieved a mean intersection-over-union (mIoU) rate of 89.04% for ARs, indicating its high accuracy in identifying these weather patterns and a superior performance than most deep learning–based models for AR detection. In addition, ARDetect can be executed faster than the TempestExtremes method (seconds vs minutes) for the same period. This provides a significant benefit for online AR detection, especially for high-resolution global models. An ensemble of 10 models, each trained on the same dataset but having different starting weights, was used to further improve on the performance produced by ARDetect, thus demonstrating the importance of model diversity in improving performance. ARDetect provides an effective and fast deep learning–based model for researchers and weather forecasters to better detect and understand ARs, which have significant impacts on weather-related events such as floods and droughts.

KEYWORDS: Atmospheric river; Algorithms; Data mining; Data processing/distribution; Data science; Deep learning

1. Introduction

Atmospheric rivers (ARs) are elongated regions of moisture-rich air that are transported from the tropics to higher latitudes. They are known to be responsible for a significant portion of precipitation in many regions and have been linked to numerous weather- and climate-related impacts, including heavy precipitation and flooding. Thus, accurately and efficiently detecting ARs is crucial for a wide range of applications, such as water resource management, as well as to study how climate change can affect these phenomena.

Climate simulations are the main way of studying Earth’s climate system and any effects that are brought about by climate change. However, each of these simulations produces a vast amount of data. For example, the current phase of the Climate Model Intercomparison Project (CMIP6; Eyring et al. 2016) is made up of hundreds of different simulations and was projected to produce 18 PB of data (Balaji et al. 2018). Unfortu-

An example of a natural disaster is a hurricane. These storms can cause significant damage and loss of life. In order to prepare for hurricanes, it is important to accurately predict their trajectory and intensity. One approach to this problem is the use of atmospheric river tracking methods. Atmospheric rivers are long, narrow, and transient corridors of strong horizontal water vapor transport that can transport large amounts of moisture to different regions. These systems are often associated with severe weather events such as floods and droughts.

ARs are typically characterized by a high concentration of water vapor and temperatures that are colder than the surrounding atmosphere. They are often associated with heavy precipitation, which can cause flooding and other types of damage. Understanding and predicting the behavior of ARs is important for disaster preparedness and response.

ARs are detected using a variety of methods, both traditional and modern. Traditional methods include visual inspection of satellite imagery and the use of atmospheric models. Modern methods, such as those based on machine learning, can process large amounts of data more quickly and accurately. Some of these methods, such as ARDetect, use convolutional neural networks to identify ARs in atmospheric data.

ARDetect is a deep learning model that has been trained to identify ARs in atmospheric data. It has been tested on ERA5 data from 1960 to 2020 and has achieved a mean intersection-over-union (mIoU) rate of 89.04% for ARs. This indicates that the model is able to accurately identify these weather patterns. In addition, the model can be executed faster than traditional tracking methods, which can improve the efficiency of detecting ARs.

The use of deep learning models like ARDetect has the potential to improve the accuracy and efficiency of AR detection. This can help to better understand the impacts of ARs on weather and climate, and to develop strategies for mitigating their effects.
typically associated with a low-level jet stream ahead of the cold front of an extratropical cyclone” (https://glossary.ametsoc.org/wiki/Atmospheric_river). It can be noted that this definition is not quantitative, so disagreements in the different algorithms occur just because of different interpretations of this definition.

However, as Lora et al. (2020) discuss, most of these methods agree on the presence of most ARs, especially strong ones and in five distinct areas over the extratropical oceans. Differences then occur on the spatial extent of weaker ARs, and this is due to the relatively vague definition of an AR. As such, ARs detected to be in the extratropics need to be considered carefully, as should those with thin, long filament-like structures as the latter are usually attributed to weak systems. These can be considered as ARs by certain algorithms and not by others depending on the use case that the original developer had in mind.

b. **AR detection using machine learning**

Algorithms using ML and DL methods have recently been developed to detect ARs in meteorological data. These are summarized below and tabulated in Table 1.

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Fields</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu et al. (2016)</td>
<td>TMQ</td>
<td>Accuracy = 90%</td>
</tr>
<tr>
<td></td>
<td>Land–sea mask</td>
<td></td>
</tr>
<tr>
<td>Kurth et al. (2018)</td>
<td>Water vapor at different altitudes</td>
<td>Tiramisu mIoU = 59%</td>
</tr>
<tr>
<td></td>
<td>Wind at different altitudes</td>
<td>Modified DeepLabv3+ mIoU = 73%</td>
</tr>
<tr>
<td></td>
<td>Precipitation at different altitudes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Temperature at different altitudes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pressure at different altitudes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Humidity at different altitudes</td>
<td></td>
</tr>
<tr>
<td>Muszynski et al. (2019)</td>
<td>TMQ</td>
<td>Accuracy = 77%–91%</td>
</tr>
<tr>
<td>Kapp-Schwoerer et al. (2020)</td>
<td>TMQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U wind speed at 850 hPa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V wind speed at 850 hPa</td>
<td>mIoU = 56%</td>
</tr>
<tr>
<td></td>
<td>Mean sea level pressure (MSLP)</td>
<td></td>
</tr>
<tr>
<td>Prabhat et al. (2021)</td>
<td>TMQ</td>
<td>AR IoU = 39%/77%</td>
</tr>
<tr>
<td></td>
<td>U wind speed at 850 hPa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V wind speed at 850 hPa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRECT</td>
<td></td>
</tr>
<tr>
<td>Buch (2021)</td>
<td>TMQ</td>
<td>AR IoU = 33%–44%</td>
</tr>
<tr>
<td></td>
<td>U wind speed at 850 hPa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V wind speed at 850 hPa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MSLP</td>
<td></td>
</tr>
<tr>
<td>Higgins et al. (2023)</td>
<td>Zonal wind at 850 mb</td>
<td>AR IoU = 20%–50%</td>
</tr>
<tr>
<td></td>
<td>Meridional wind at 850 mb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Surface pressure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IWV</td>
<td></td>
</tr>
<tr>
<td>Tian et al. (2023)</td>
<td>Zonal wind at 850 mb</td>
<td>AR IoU = 38.5%</td>
</tr>
<tr>
<td></td>
<td>Meridional wind at 850 mb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IWV</td>
<td></td>
</tr>
</tbody>
</table>

Muszynski et al. (2019) developed a two-stage method that first finds possible ARs using topological data analysis and then uses a support vector machine (SVM) to determine whether each of these candidates is a true AR, with the ground-truth labels coming from the Toolkit for Extreme Climate Analysis (TECA; Prabhat et al. 2015). This method is applied to the total precipitable water (TMQ) field, but the authors argue that it could be applied to many other fields of interest. Having applied this method to a number of different datasets, each having different spatial and temporal resolutions, the method obtained an accuracy between 77% and 91%.

Another detection-based method was developed by Liu et al. (2016). The authors created a 4-layer convolutional neural network (CNN)-based classifier and used data from cropped sections of the globe to infer whether a tropical cyclone (TC) or an AR is present. The network used TMQ and a land–sea mask as inputs and the required labels were also derived from TECA. After training, the network obtained an accuracy score of 90% for detecting ARs.

Besides detection, deep learning has also been employed to perform image segmentation for ARs. In this case, the desired
output of the network is an array similar to that output by TECA or any other heuristic that shows whether each position in the image belongs to an AR. This is more useful than simple classification as the spatial attributes of features, ARs in this case, can be more easily studied. One useful metric used to quantify how well the inference produced by the network matches the given label is the intersection-over-union (IoU). The IoU measures the common area (area of intersection) between the given label and the inference given by the network and divides it by the total area of the combined label and inference. The mean intersection-over-union (mIoU) is then the mean of the IoUs obtained for each test case. Hence, an mIoU value as close as possible to 100% is desired.

Kurth et al. (2018) repurposed the Tiramisu (Jégou et al. 2016) and DeepLabv3+ (Chen et al. 2018) networks for image segmentation to detect ARs. They used a dataset containing various different variables including water vapor, wind speed, precipitation, temperature, pressure and humidity at different heights in the atmosphere for the whole globe. After training the two different networks, the Tiramisu network obtained a mIoU for ARs of 59% and the DeepLabv3+ based network obtained an mIoU of 73%.

Prabhat et al. (2021) built on this work by using a similar adapted DeepLabv3+ network to perform image segmentation for ARs. The dataset the authors used also spanned the whole globe, but they only included four variables in their dataset: TMO, meridional wind speed at 850 hPa, zonal wind speed at 850 hPa, and the total precipitation rate (PRECT). The authors also had two different labeling sources: one was TECA while the second source was a set of 459 samples that were labeled by human experts. They noted that the mIoU for ARs in the expert labeled dataset between multiple experts is 34%, showing that even experts can disagree significantly on the spatial extent of an AR. When training their network on labels obtained from TECA, the authors reported an mIoU for ARs of 77% between the inferences made by the network and the labels obtained from TECA. However, when training using the expert-created labels, the AR mIoU dropped to 34%, possibly suggesting that the uncertainty between the definition of an AR across the experts may be hindering the performance of the network.

Kapp-Schwoerer et al. (2020) used the expert-labeled dataset of Prabhat et al. (2021) and network based on CGNet (Wu et al. 2021), a context-guided deep neural network for image segmentation, to perform this task for ARs and TCs. While the authors did not give an mIoU for ARs only, the mIoU for ARs, TCs, and background features is that of 56%. Finally, Buch (2021) used the same dataset and another adaptation of CGNet to attempt to produce a better performing network. The one difference between the two methods is that the dataset for the second method was restricted to only four areas instead of the whole globe. These were the northeastern Pacific (NEPAC), the southeastern Pacific (SEPAC), the North Pacific (NPAC), and the South Pacific (SPAC). It should be noted that NEPAC and SEPAC are mostly contained in SPAC. The authors showed that the network trained using all four regions obtained an all-class mIoU of 55% when testing on NEPAC data. Furthermore, a NEPAC-trained network obtained an all-class mIoU of 57% when trained and tested on NEPAC-only data, showing that a region-specific network might be useful.

Higgins et al. (2023) trained a network based on CGNet (Wu et al. 2021) on data spanning different regions, different horizontal resolutions and different climate models. The authors argued that IVT was computationally expensive to calculate and that some model outputs do not have this field present. As such, they used zonal wind at 850 mb (1 m/s = 1 hPa), meridional wind at 850 mb, surface pressure, and total column integrated water vapor (IWV) derived from specific humidity as inputs, with the aim of making the resultant network more flexible to preexisting datasets. However, when evaluating their results against labels created by 8 different algorithms from the ARTMIP project, the AR IoU values ranged from 20% to 50%.

Tian et al. (2023) employ an ensemble of 20 different deep learning models to perform image segmentation for ARs. Similar to the previous method, the authors do not utilize IVT as an input variable to their models, but instead only use zonal wind at 850 mb, meridional wind at 850 mb, and IWV. Each model in the ensemble was trained independently, but after training, the final result was obtained via majority voting. Therefore, the class of a certain position is determined by which class most models in the ensemble assign the position to. When testing across the whole test dataset used in their study, the ensemble of models obtained an mIoU for ARs of 40.5% and the authors showed that the ensemble performed much better than any single model in the ensemble, with none surpassing an mIoU of 38.5%.

As can be noted, the best mIoU obtained was that of 77% and the best accuracy was that of 91%. In this study, we present a deep learning model aimed at image segmentation for ARs in climate data. This model obtains an mIoU of nearly 90%, an accuracy of 99.41% and a recall rate of 90.34%. This improvement in performance is attributed to a larger training dataset and more fields than used previously as well as a network architecture specifically tuned for the application at hand.

Section 2 details the architecture of this model and the dataset used. Section 3 presents the results obtained from this model and discusses why the fields used in this study were chosen; it also shows how creating an ensemble of this model improved the overall performance. Finally, section 4 summarizes the study’s results.

2. Deep learning model

This section describes the deep learning model, referred to as ARDetect, being presented in this study. Details on the structure of ARDetect, including any hyperparameters used, are given as well as a description of the data used as inputs and the expected output.

a. Data

ARDetect employs several meteorological fields to identify ARs. These fields are treated as separate channels in an image, and a binary mask is generated as output to indicate whether a given pixel belongs to the AR class. The input
fields consist of IVT, total column water (TCW)—the vertical integral of water vapor, cloud liquid water, and cloud ice in a single column—and wind speed at 850 hPa. In addition, the same fields and the AR mask from the previous time step are included as inputs. These fields were decided after performing a feature importance process, which is detailed in section 3b, on a larger set of fields. The ERA5 reanalysis dataset (Hersbach et al. 2018, 2020) is used to obtain the meteorological data, which provides a comprehensive representation of past climate conditions through simulations and observations. We use 3-hourly data, and they were interpolated from the native resolution of 0.25° to a resolution of 0.5° horizontally. This was done as a better performing version of ARDetect was obtained with the coarser resolution, possibly due to implicit filtering of noise in the input data. The resulting input and output arrays were each 360 rows by 720 columns. Finally, the dataset was divided into three subsets for training, validation, and testing. spanning 1960–2010, 2010–15, and 2015–20, respectively.

A way to obtain the labels, that is, AR masks, for ARDetect was required. TempestExtremes was used for this purpose. The method used to create these AR masks primarily followed that used by Ullrich et al. (2021):

- Detection of tropical cyclones: First, plausible candidates for TCs are identified as local minima in the mean sea level pressure field and then narrowed down using two criteria that check for a warm-core structure. Then the detected TCs are joined over time to create tracks, with certain criteria used to remove false detections. Finally, the size of the TCs is defined by calculating the radius at which wind speed drops below 8 m s⁻¹.

- Detection of AR candidates: ARs are identified as ridges in the IVT field. This is calculated from the eastward and northward components of the IVT. The Laplacian using 8 radial grid points at 10° great circle distance (GCD) is then calculated, and AR candidates are designated as those ridges with a Laplacian value smaller than −2 × 10⁶ kg m⁻² s⁻³ rad⁻². Then, any AR areas that are within 15° of the equator are cropped, and those that have an area smaller than 4 × 10⁶ km² are removed. The former is done to remove any MCS-like systems, while the cropping is performed as ARs are considered to be midlatitude features.

- Final AR masks: TCs can sometimes exhibit similar features to ARs in the IVT field since it is necessary to remove TCs from any of the AR candidates. Hence, any AR candidates that are within 8 GCD of any TC center as detected previously are removed from the dataset. This final dataset contains all AR masks that are to be considered as labels for ARDetect.

Given that the method used to obtain these AR maps is crucial to the performance of ARDetect, it would be important to address the robustness of these maps, that is, whether any biases are present that would make the resultant ARDetect more performant in a specific area. As discussed above, TempestExtremes is an algorithm that relies on values of IVT and we consider the maps produced as truth in our case. As such, the algorithm itself does not have biases, but the underlying IVT field might have. Since we are using ERA5 reanalysis data, the quality of the data very much depends on the observational dataset used during the compilation of ERA5. This means that the quality of data in the NH would be superior to that of the SH before satellite data were available (pre-1979), but no differences should be seen after. Also, given the large dataset constructed for training ARDetect, pre-1979 data constitute a minor part of the whole dataset, so we do not see any biases between labels in the NH and SH.

It should be noted that if the AR maps are produced using some algorithm other than TempestExtremes, or with TempestExtremes but using different settings, it is expected that ARDetect will mirror those labels. As such, a possible extension of this work is to use uncertainty quantification to produce a map of where an AR might be, which will help bridge some of the differences between the different definitions.

Figure 1 shows an example of the collected meteorological fields and associated AR mask for one time step. These fields are then collected in such a manner that each input to ARDetect has fields from two consecutive timesteps stacked together. Each input feature was then standardized using the minimum and maximum value for that feature for that input case.

b. Architecture

The ARDetect model utilizes a U-Net-style architecture (Ronneberger et al. 2015) with multiple convolutional layers. Automatic hyperparameter tuning was performed using the Ray Tune (Liaw et al. 2018) Python package to determine the optimal architecture. Initially, random searches were executed to narrow down the search space to a range that included the highest performing architectures. Bayesian optimization was then employed to arrive at the best architecture, which is shown in Fig. 2. In Bayesian optimization, the user first defines a starting search space. The algorithm then refines this iteratively depending on the results of any previously executed searching. This is done by fitting a Gaussian process regression model on the search space to attempt to narrow it down. Points from this search space are then selected, and a cost function, usually taken to be the performance of the model, is computed. Then, the next iteration of search space refining is done. The process ends after a set number of iterations. Details of the final model structure are provided below.

As previously stated, the model takes inputs with a shape of 360 rows and 720 columns, with a batch size of 16 per graphics processing unit (GPU) resulting in an input shape of (16, 360, 720). The inputs are processed through four convolution blocks, each comprising a convolutional layer, a batch normalization layer, a dropout layer, and a leaky ReLU activation function. The dropout rate used was 0.12%.

The four convolution blocks had 6, 24, 96, and 384 feature maps, respectively, and used a kernel size of 4. After processing through a block, a copy of the outputs was retained to serve as connections to the upsampling branch of the U-Net. Additionally, a maxpooling operation was applied to reduce the input size by half in preparation for the next convolution block.
After passing through the downsampling branch of the U-Net, the resulting latent space is fed into the upsampling branch. Each block in this branch used the outputs from the previous layer as inputs and employed a ConvTranspose2D layer, a dropout layer, and leaky ReLU activation to upsample the inputs. The outputs were concatenated with the corresponding connections originating from the downsampling branch of the U-Net. The concatenated data were then used as inputs to a convolutional layer to prepare the data for the next upsampling block. The upsampling blocks used the same number of feature maps as in the downsampling branch and padding was used throughout the network to ensure compatibility of sizes for the inputs and outputs.

A final convolutional layer was used to transform the resulting data such that two channels were outputted. This was done so that the CrossEntropy loss function could be used. The network weights, of which there are around 63 million, were initialized using the Xavier method (Glorot and Bengio 2010). RMSProp was selected as the optimizer with a learning rate of $1.39 \times 10^{-3}$, an alpha value of $5.65 \times 10^{-1}$, and an eps value of $8.34 \times 10^{-2}$. In our dataset, 95.5% of the pixels were non-AR pixels and 4.5% were AR pixels. Therefore, due to the imbalanced nature of the training dataset, class weights were supplied to the optimizer to weight the contribution of each example accordingly.

c. Practical considerations

ARDetect has a large number of parameters. That, as well as the large data size required that distributed training be used to perform training in a reasonable timeframe. For this, one node having two Tesla P100 GPUs, each with 16 GB of RAM and 32 physical cores, was used. The whole dataset, including the training, validation and testing datasets, amounted to around 1.4 TB of data, so a dataloader was used to generate the inputs as required by training. Python 3.10 with PyTorch 11.3 was used as the base environment and other Python packages were used to create the data. Using this setup, ARDetect was trained for 24 epochs in around 72 h. However, it only took 0.53 s to produce an inference using a CPU or 0.014 s using a GPU when averaging over 100 examples. This is a major improvement on TempestExtremes that takes minutes to first detect TCs then detect ARs and finally produce AR maps. This provides a significant benefit for online AR tracking especially for high-resolution global models. In such an application, at the end of each simulated time step, the required data would be collected and passed onto ARDetect, which will make its prediction. Then, if the user wants AR masks to be part of the climate model output, the masks are pushed to the climate model’s output system. However, it should be noted that the AR map for the initial time step would need to be provided from TempestExtremes.

3. Results and discussion

The resulting deep learning model, ARDetect, was evaluated using the test set described above. The inferences obtained were also investigated to understand how the model generates its results.
a. Model performance

ARDetect performed well in creating a mask to show the position of any atmospheric river present in meteorological data, with an mIoU of 89.04% for the AR class in the testing dataset. Figure 3 shows two examples of predictions made, indicating that the major features of ARs are well defined with minor disagreements between the predictions and labels occurring for some thin AR regions. This is also expected of human experts, as discussed previously in section 1a, so the replication of this by ARDetect was not unexpected.

The confidence matrix for each pixel in each image in the testing dataset was also calculated and is shown in Table 2. The matrix indicates that the vast majority of pixels (99.41%) were classified correctly, with the majority of these pixels on the inverse diagonal (top-left to bottom-right). Both the false positive rate (0.08%) and the false negative rate (0.51%) are very low compared to the true positive (4.77%) and true negative (94.64%) rates, indicating the superior performance of ARDetect.

Using the values in the confidence matrix, accuracy, precision, and recall rates were calculated. ARDetect has an accuracy of 99.41%, while the recall rate is 90.34% and the precision rate is 98.35%. These values show that ARDetect is achieving high performance for image segmentation of ARs.

The mIoU for ARs was also calculated for both the latitudinal and longitudinal directions to identify where ARDetect is more likely to produce erroneous predictions. In the top panel of Fig. 4, we show how the mIoU for ARs varies across different latitudes. The value of mIoU is zero in the tropics due to TempestExtremes being set up in such a way that no ARs are to be detected between 15°N and 15°S as discussed in section 2a. It could be noticeable that ARDetect does very well in the midlatitudes and nearer to the equator to a lesser extent, while it performs less well at the poles. The poorer performance nearer the poles is expected as fewer ARs are present in these areas, so a limited number of training data are available. This makes it harder for ARDetect to obtain maximal performance in these regions. It could also be noted that ARDetect performs marginally better in the Northern Hemisphere with an average mIoU of 49% across all latitudes versus the Southern Hemisphere, which had an average mIoU of 48.3%. However, the highest mIoU in the Northern Hemisphere of 89.7% was surpassed by an mIoU of 93% in the Southern Hemisphere. Similarly we show how the mIoU for ARs varies across different longitudes in the bottom panel of Fig. 4. This shows that ARDetect performs marginally better in the Western Hemisphere with higher mIoU values (average mIoU: 82.1%; maximum mIoU: 87.6%) than in the Eastern Hemisphere (average mIoU: 77.6%; maximum mIoU: 84.2%). It can also be noted that the mIoU drops to zero at the date line. This is because zero padding was employed as the last layer of ARDetect to ensure common sizes between inputs and outputs. As such, no ARs will be detected in the last 1.5° of latitude by the current iteration of ARDetect, but it is an issue to be rectified in any future work.

b. Feature importance

In the field of deep learning, the selection of relevant features, meteorological fields in our case, to include as inputs is

![Fig. 2. Architecture of ARDetect.](image-url)
a crucial aspect in model building. One method to determine the importance of these features is through the use of feature importance analysis. In this study, we employed this technique to determine the most important meteorological fields to be included in the final ARDetect model. We used the methodology described by Breiman (2001):

- Training of deep learning model: A deep learning model was trained using a training dataset. The network used was one that had acceptable performance for the task. As such, it only had three hidden layers and considerably fewer parameters than the final ARDetect architecture.
- Base performance calculation: A testing dataset was used to obtain the base performance of the deep learning model.
- Permutation of single feature: A single feature was permuted across all test cases, and the performance of the deep learning model was calculated using the altered dataset.
- Performance difference calculation: The difference between the performance using the permuted dataset and the base performance was calculated.
- Obtain difference for each feature: The previous two steps were repeated for all features present in the dataset, such that the difference in performance was calculated for each input feature.
- Obtain list of most important features: The differences were sorted in descending order to determine the most important features.

This process was carried out using the validation set that was described in section 2a. The training set for this process spanned from 2010 until 2012 (i.e., 3 years of data) while the testing set was selected to have data from 2014 and 2015 (i.e., 2 years of data). Table 3 shows the variables included in this process. A time dimension was also added by including the same fields from the previous time step to the one being considered as inputs.

The deep learning network used to perform this feature importance process was a 3-layer U-Net with each layer having 37, 74, and 148 filters, respectively. The AdamW optimiser was used together with CrossEntropyLoss. Due to the large imbalance of AR pixels and non-AR pixels, weights amounting to the percentage of AR and non-AR pixels (4.04% and 95.96%, respectively) were passed to the loss function so that AR pixels were given more importance during training. This network obtained an mIoU for ARs of 84.05%.

After training the network and performing the feature importance process, those variables having a positive influence on the performance of the network were carried over to the development of ARDetect. The full results are plotted in Fig. 5.
The results showing the variables that improved the network’s performance are tabulated in Table 4. The mean AR mIoU in this table is the mean performance of 30 bootstrapped testing datasets with the variable in question being shuffled. The difference column of Table 4, which is visualized for all the variables tested in Fig. 5, is the difference between this mean AR mIoU and the performance of the model when testing using the original testing dataset. These show that the IVT field is the most important variable for the network to make its inferences. Other important variables are total column water and wind speed at 850 hPa. The AR mask from the previous time step is also important, presumably as it is a good starting point for the inference on the current time step. It was also noted that adding a time dimension to the inputs, that is, adding the same fields from the previous time step, helped improve performance.

c. Effect of size of training dataset

One important aspect to the performance achieved by ARDetect is the size of the training dataset. As noted before, the previous studies noted in section 1 had small datasets relative to the one used in this study. As such, we conducted a test to confirm that the size of the training dataset is indeed an important aspect of this study.

This test was performed by training versions of ARDetect using only subsets of the original training dataset. The subsets ranged between 10% and 100% of the original dataset size. To ensure accurate results, this was repeated 10 times, with different starting weights, that is, utilizing random weight initialization, each time. By using random weight initialization, we were intending to capture the uncertainty due to initial conditions.
Figure 6 shows a boxplot of these results and a regression line for the means in a red dotted line. It can be noted that this line is increasing with a larger training dataset, showing that the size of our training dataset is helping ARDetect achieve the performance it does. However, there is a large decrease in performance when using 30% of the dataset. This could be attributed to higher amounts of noise injected into the dataset as more examples were added. Despite this decrease, an increasing trend line would still be obtained if this data point is not included, even if the trend would be slight. Nevertheless, ARDetect can still reach 81% mIoU even with 10% of dataset used, which is still far superior than other DL AR tracking models, showing that the choice of input fields is very important.

d. Ensemble of models

Ensemble modeling is a powerful technique that involves combining the predictions of multiple models to improve overall performance. In this section, we describe our ensemble approach for predicting the positions of ARs in meteorological data.

To create the ensemble of models, we trained the ARDetect architecture 10 times using the same training dataset but different initial random seeds. This ensured that the initial weights of the models and the batches of data used for training were different for each run. To obtain a single prediction from the ensemble, the outputs of each trained version of ARDetect were first obtained. Each inference has values ranging from 0 to 1, denoting the probability of a pixel belonging to an AR (0 being 0% and 1 being 100%). Then, any values below 0.5, or 50%, are set to 0, and any values larger than 0.5 are set to 1. Once this is done for the outputs of all members of the ensemble, the average is calculated. Once again, any values below 0.5 are set to 0, and any values larger than 0.5 are set to 1, thus obtaining the average mask from the ensemble.

After training, we collected the predictions of each model and averaged the masks produced by each model. As such, there were 10 masks produced for each time step of the training dataset (one from each model), and then an average of these masks was calculated. We then calculated the final statistics using the average mask.

The ensemble of models achieved an mIoU for ARs of 89.95%, which is a 0.94% improvement from the singular ARDetect model. It should be noted that none of the ensemble members could produce better performance than the ensemble, showing that the ensemble method provides an improvement in performance, as was also found by Tian et al. (2023). This improvement could be attributed to the different models having slightly different end point when converging due to their starting points, thus capturing the loss surface better.

The confidence matrix for each pixel in each image in the testing dataset using the ensemble predictions was also calculated and is shown in Table 5. The matrix indicates that the vast majority of pixels (99.47%) were classified correctly, with the majority of these pixels on the inverse diagonal (from top left to bottom right). ARDetect tends to overpredict ARs, as the false positive rate (0.46%) is slightly higher than the false negative rate (0.08%). However, both rates are very low compared to the true positive (4.78%) and true negative (94.64%) rates, indicating that ARDetect is performing well.

Using the values in the confidence matrix, accuracy, precision, and recall rates were calculated. ARDetect has an accuracy of 99.47%, while the recall rate is 91.22% and the precision rate is 98.35%. These values show that the ensemble of ARDetect models is improving on the singular model by producing fewer false negatives, and more true negatives, which results in a higher performance.

Table 4. Feature importance results showing only those variables that improved the network performance. The mean AR mIoU is calculated from 30 different trained networks and the difference from the original trained network is also shown.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mean AR mIoU</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current time step IVT</td>
<td>5.69%</td>
<td>78.36%</td>
</tr>
<tr>
<td>Previous time step IVT</td>
<td>60.51%</td>
<td>23.54%</td>
</tr>
<tr>
<td>Previous time step AR mask</td>
<td>64.96%</td>
<td>19.08%</td>
</tr>
<tr>
<td>Current time step TCW</td>
<td>79.94%</td>
<td>4.11%</td>
</tr>
<tr>
<td>Current time step wind speed at 850 hPa</td>
<td>83.19%</td>
<td>0.86%</td>
</tr>
<tr>
<td>Previous time step TCW</td>
<td>83.81%</td>
<td>0.23%</td>
</tr>
<tr>
<td>Previous time step wind speed at 850 hPa</td>
<td>84.02%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

Table 5. Confidence matrix for ensemble results.

<table>
<thead>
<tr>
<th></th>
<th>Predicted AR</th>
<th>Predicted non-AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual AR</td>
<td>180 685 666  (4.78%)</td>
<td>17 316 016  (0.46%)</td>
</tr>
<tr>
<td>Actual non-AR</td>
<td>2 892 119  (0.08%)</td>
<td>3 581 352 599  (94.69%)</td>
</tr>
</tbody>
</table>
4. Summary

In this study we developed a deep learning model, termed ARDetect, for atmospheric river (AR) image segmentation in meteorological data. ARs are long and narrow regions of high moisture content in the atmosphere that can cause extreme precipitation, leading to floods and other severe weather events. The accurate detection and segmentation of ARs is crucial for improving our understanding of their behavior, and for forecasting and monitoring their impacts.

ARDetect was trained on the ERA5 dataset, with data covering the period from 1960 to 2020. The labels used to train the model were obtained from the TempestExtremes algorithm, which identifies and tracks ARs in atmospheric data. The model’s architecture was found using automatic hyperparameter tuning, resulting in a high performing model. Feature importance was used to identify the most significant input variables, with integrated water vapor transport (IVT) and total column water (TCW) being the most important meteorological features that were used as inputs. To include the time-dependent nature of ARs, ARDetect also took in these variables for the previous time step to that being considered as inputs. The TempestExtremes AR mask for the previous time step was also found to be an important input field when feature importance was carried out.

The performance of ARDetect was evaluated using testing data, and it achieved an accuracy of 99.41%, a recall of 90.34%, and a precision rate of 98.35%. The model’s overall performance was evaluated using the mean intersection over union (mIoU), and it achieved an mIoU of 89.04% for the AR class, superior to other deep learning models developed for this task. Despite the high performance, most of this is seen in the midlatitudes (30°–60°) as opposed to latitudes higher than 60°. Interestingly, ARDetect was found to work marginally better in the Northern and Western Hemispheres than in the Southern and Eastern Hemispheres. It should also be noted that the performance of the model as given here only applies to the case when using the maps generated by TempestExtremes as described in this study. If a different labeling system is used for testing, it is likely that ARDetect will obtain inferior performance, but this should be recaptured if the labeling system during training is changed accordingly. This discussion extends to any human-generated labels due to inconsistencies between different human labelers. As Prabhat et al. (2021) show, there is only a 35% mIoU between different human labelers.

To improve the model’s performance even further, an ensemble of ARDetect models was created using the same dataset but different initial weights. The ensemble produced an mIoU of 89.95%, which slightly outperformed the singular model. This approach demonstrates the effectiveness of model ensembling in improving the overall performance of the model.

Overall, ARDetect is a powerful deep learning model that shows great potential for improving our understanding of atmospheric rivers and their impacts. The model’s ability to accurately detect and segment ARs could have applications in weather forecasting, climate modeling, and extreme weather event detection. Future work could involve exploring different datasets and including ARDetect in a climate model to perform online tracking with a view to providing a better understanding of ARs, as well as using more temporal information as inputs.

Acknowledgments. We would like to acknowledge Hsiang-He Lee for help in reviewing the manuscript prior to submission. This work was supported by Lawrence Livermore National Laboratory Directed Research and Development project 22-ERD-013 and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was released under LLNL-JRL-849171.

Data availability statement. ERA5 pressure-level data (Hersbach et al. 2018, 2020) are available for download from ECMWF’s Copernicus Climate Change Service (C3S) Climate Data Store (CDS) at https://doi.org/10.24381/cds.bd09156. Similarly, ERA5 single-level data are available for download from ECMWF’s C3S CDS at https://doi.org/10.24381/cds.adbb2d47. TempestExtremes (Ullrich et al. 2021) is available at https://github.com/ClimateGlobalChange/tempestextremes.

REFERENCES


