Universiteit Utrecht

Master Applied Data Science, Utrecht University

River Morphology Classification Using Deep Convolutional Neural Networks

June 30, 2024

Author:
Mara van Boeckel (1385747)

Supervisors:
Dr. Daan Beelen
Dr. ir. Jaap Nienhuis
Niek Collot d’Escury MSc

Abstract

River morphology classification has been researched since the 19th century. With the rise of higher-resolution satellite imagery and the development of more advanced machine learning algorithms, automated and procedural classification can be employed. This thesis investigates to what extent a (Deep) Convolutional Neural Network ((D)CNN) can be applied on satellite imagery to perform river morphology classification and identifies the optimal architecture with a limited dataset. This research represents the first attempt at using deep learning methods to classify river morphologies and demonstrates the feasibility and potential of DCNNs.
Several experiments are employed to find the most suitable model for river classification. The optimal architecture is a two-input DCNN utilizing satellite images and centerlines of the rivers of the SWOT River Database as input, having five convolutional and pooling layers, with 24, 48, 96, 192, and 384 feature maps in the convolutional layers. It includes a fully connected layer with 64 dense units, dropouts of 0.2 after each pooling layer and between the fully connected- and the output layer, and batch normalization. The weighted F1-score is 53% but the network still exhibits slight overfitting despite the use of regularization techniques. The final model was able to classify the four different river morphologies, anastomosing, braiding, meandering, and wandering, with an F1-score of 25%, 49%, 37%, and 65%, respectively. The DCNN significantly outperforms a baseline that predicts the majority class. A Cohen's d value of 0.9 demonstrates the DCNN's large improvement over the baseline. The model has difficulty distinguishing between anastomosing and braided and meandering and wandering which is to be expected since human classifiers also perceive this difficulty. Additionally, most river segments have multiple labels in one image, which makes the classification challenging. This study was the first to explore and demonstrate the feasibility and potential of using DCNNs to classify river morphologies. Gathering more labeled data is necessary for further improvement of the network's performance.

Keywords: Deep Convolutional Neural Network, DCNN, deep learning, image classification, river morphology classification.

Table of content

List of Figures

List of Tables

List of Equations

Glossary

Backpropagation
To learn the optimal filter structure, the network is optimizing the weights of all its connections to improve task performance based on previous outcomes. This is crucial for optimizing the performance of the network. This algorithm is called backpropagation.

Batch size
Refers to the number of data points (a subset of the training data) used in each iteration of training.

Convergence
Refers to the stabilization of the model's performance metrics such as loss or accuracy. It indicates that the model has learned as much as possible from the data.

Epoch
Refers to one pass through the complete training set during training. 1 epoch means the model has seen the entire training dataset once.

Gradients
Represent the rate of change of the loss function with respect to the parameters of a network.

Learning rate
A hyperparameter that controls the size of the steps taken during the optimization process of training a model. It determines how much the model's parameters are adjusted with respect to the loss gradient during training.

Optimizer
An optimizer is used to minimize the loss function by adjusting the model's parameters during training.

Overfitting
When the model learns the patterns in the training data too well it negatively impacts its ability to generalize to unseen data. The model captures noise or random variations in the training data rather than the underlying pattern.

Padding
Padding is used to control the spatial size of the output feature maps. It involves adding extra pixels around the input image. When padding='valid': No padding is applied. The output size is smaller than the input size. padding='same' means padding is applied so that the output size is the same as the input size.

Patience
The number of epochs with no improvement after which training will be stopped.

Underfitting
When the model is too simple to capture the underlying patterns in the data.

1. Introduction

The rivers across the world have a wide variety in size, shape, dynamic, and characteristics. Rivers and river deltas play a vital role in ecosystem support, sediment transport, water quality regulation, and climate regulation (Manning, 2022). The dynamics of rivers are also significantly influenced by human activities and climate change (Li et. al., 2022). Global labeling of river morphologies therefore gives valuable insight into understanding and managing different types of rivers, which can be a significant advance in fluvial geomorphology. River morphology refers to a river's shape, sediment movement, and erosion patterns (Nath & Ghosh, 2022) and is influenced by several variables (Lancaster & Bras, 2002) also called fluviomorphological parameters (FMPs). River morphology classification is therefore frequently executed using attributes such as river width, sinuosity of the channel, channel form, slope, meander wavelength, or centerlines (Leopold & Wolman, 1957; Alabyan & Chalov, 1998; Moraes Frasson et al., 2019; Li et al., 2022). As satellite imagery improves in coverage and resolution, satellite data can be used to create a global classification atlas for all river morphology classes. According to Nyberg et. al. (2023), machine learning can be applied to identify different river shapes and patterns. Machine learning has the potential to shed a whole new light on longstanding research that has persisted since the 19th century (Reynolds & John Emslie 1851; Leopold & Wolman, 1957; Morisawa, 1989). This is because deep learning models rely solely on satellite imagery and the labels of the rivers to identify river patterns without requiring the characteristics of the rivers such as river width, slope, or flow.

This research considers the four most acknowledged river types: anastomosing, braiding, meandering, and wandering. These types can be distinguished based on stream characteristics such as the sinuosity of a channel, the number of channels, or the presence of river islands. According to Schumm (1985), anastomosing channels are distinct from the other types because they are multiple-channel systems having major secondary channels that separate and rejoin the main channel to form a network. The other three morphology types have a single channel. Schumm (1985) describes that a braided river is divided by islands or bars, where islands are relatively permanent and bars are sand-and-gravel deposits. In addition, Charlton (2008) mentions that braided channels are highly dynamic due to the frequent shifts and changes in the positions of islands and bars within the channels. They tend to have wider and shallower channels than meandering rivers. Meandering rivers have a degree of sinuosity that can vary greatly and evolve over time (Charlton, 2008). Lastly, the wandering river can have a degree of sinuosity within it but without a regular pattern like the meandering river. Examples of the four morphology types are shown in Figure 1.

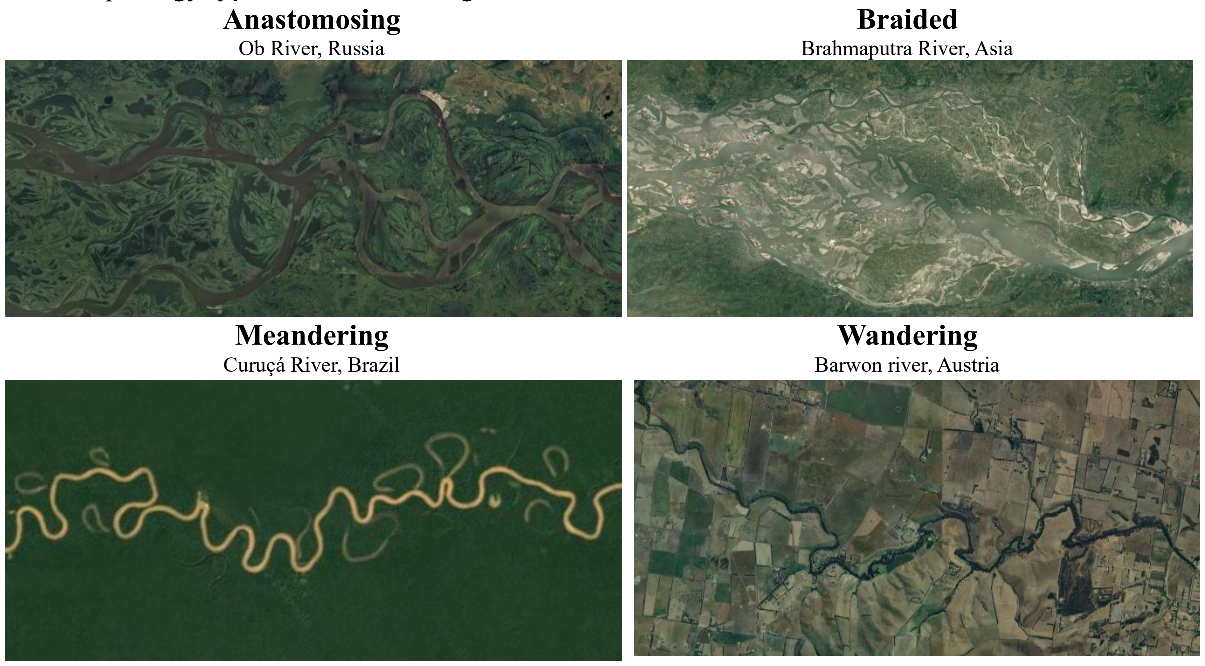


Figure 1: Example images of the four studied river morphologies (Google Maps, n.d.-a, n.d.-b, n.d.-c, n.d.-d).

1.1 Literature overview

Employing deep learning methods for river classification represent a novel approach in the field. The research paper of Nyberg et al. (2023) presents a groundbreaking global scale analysis using a pattern recognition algorithm to quantify the extensive surface area of river channel belts. Nyberg et al. applied a pre-trained Deep Convolutional Neural Network (DCNN) called VGG-19 to predict whether the river has a multi-threaded planform or a single-threaded planform based on the surface area of river channels associated with the channel belts. The model does not show significant performance, achieving an accuracy of 52% for classifying river channels into multi-threaded planforms and 48% for single-threaded planforms. In contrast, these results highlight the importance of considering channel belts in managing freshwater resources effectively, understanding river evolution, and predicting river behavior. (Nyberg et al., 2023)

Deep learning has been successfully applied in various geospatial classification tasks. Regarding river classification, Oga et al. (2020) studied the use of pre-trained CNNs, including AlexNet, to classify the river being a flood risk or not based on images of surveillance cameras. According to Oga et al., the ability to accurately classify river images captured during heavy rain or flood warnings and under normal conditions is successful, evidenced by experimental results. The utilization of satellite imagery is frequently applied for creating CNNs in geospatial studies. Adegun et al. (2023) utilized remote sensing images to classify a wide range of features present in these images using diverse pre-trained DCNNs. These models have high performances which demonstrate the feasibility of CNN-based models in earning the complex and in-homogeneous features of high-resolution remote sensing images (Adegun et al., 2023). In addition, Yao et al. (2019) introduced a new network, called the dense-coordconv network, for semantic segmentation in high-resolution remote sensing images, to enhance spatial features in land-use classification using DCNNs. The model significantly outperforms other DCNNs like U-net and Deeolab, achieving an overall accuracy of 89.48% and an 86.89% mean F1-score (Yao et al., 2019). Lastly, Pratiwi et al. (2021) also created different CNNs, including the pre-trained network AlexNet, to classify eight land use classes based on the same images Adegun et al. (2023) employed. The use of a pre-trained network in this case resulted in overfitting which is why a simpler CNN was created with a 95.2% accuracy.

1.2 Research question

This research aims to explore the feasibility and efficiency of a (D)CNN-based approach to distinguish the four different river morphologies using satellite-based datasets of the Amazon and the Himalayas. This thesis contributes as a sub-project for a larger-scale project that classifies rivers worldwide using FMPs. These two methodologies and outcomes can be compared in the future.
The main question of this thesis:

To what extent can a (Deep) Convolutional Neural Network based approach, using river satellite-based datasets, accurately classify four different river morphologies, and what is the most optimal model architecture for the classification task?

The project outline is illustrated in Figure 2 on page 3.

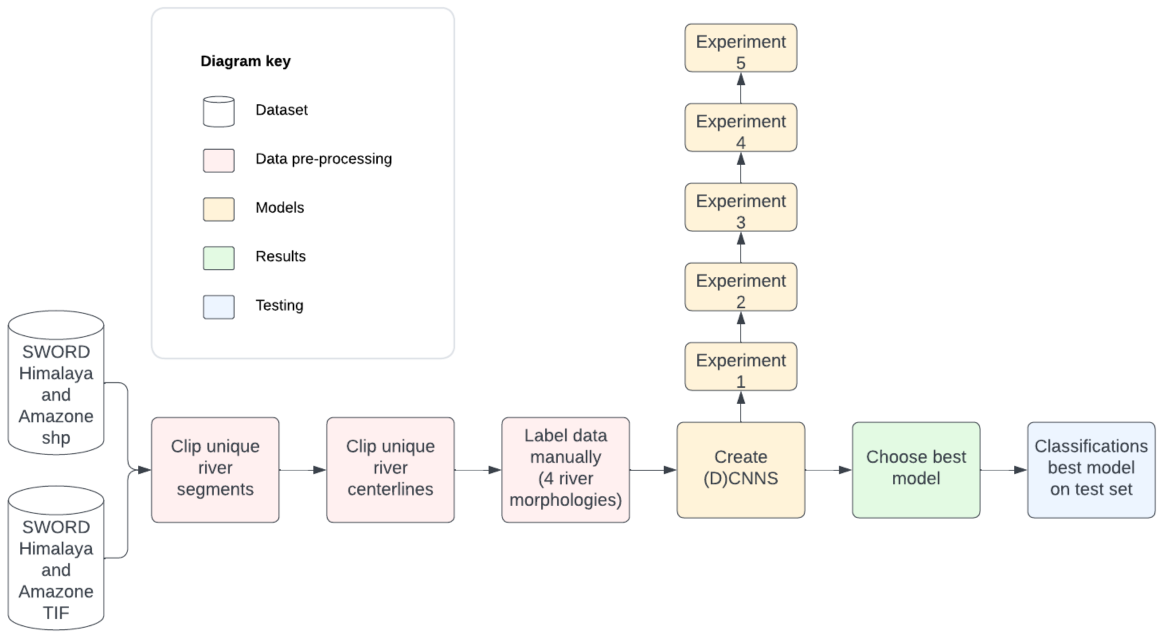


Diagram key
Dataset
Data pre-processing
Models
Results
Testing
SWORD Himalaya and Amazone shp
SWORD Himalaya and Amazone TIF
Clip unique river segments
Clip unique river centerlines
Label data manually (4 river morphologies)
Create (D)CNNS
Experiment 1
Experiment 2
Experiment 3
Experiment 4
Experiment 5
Choose best model
Classifications best model on test set

Figure 2: Project outline.

This paper is organized as follows: Chapter 2 provides insight into the data, including the data description and pre-processing. Chapter 3 outlines the methodology applied to develop the models. Subsequently, Chapter 4 presents the results of the models, compares the different performances, and shows the best model to perform the final classifications. Finally, the discussion and conclusions can be found in Chapters 5 and 6.

2. Data

This chapter provides insight in the used data and the pre-processing that has been applied to prepare for the models.

2.1 Data collection

The data used in this thesis is sourced from the Surface Water and Ocean Topography (SWOT) satellite launched in December 2022 (Biancamaria et al., 2016). The SWOT River Database (SWORD) (Altenau et al., 2021) integrates multiple satellite- and river related datasets to define reaches and nodes constituting river vector data products. SWORD data has a high resolution along with a consistent topological system for global rivers that are 30 meters wide and larger. The satellite image captures dry land as black and water as white. A TIF format of the satellite images, shapefiles containing the centerlines and additional information about the rivers are combined to predict morphologies.

2.2 Data pre-processing

The study areas in this research are sections of the Himalayas in Asia and the Amazon in South America. The shapefile contains multiple features such as river water surface elevation, width, slope, etc. The key variables in this research are *reach_id, lakeflag,* and *type.* To ensure the data only contains rivers, the variable *lakeflag* is set to zero and *type* to one, corresponding to rivers. Figure 3 illustrates the rivers in the study areas.

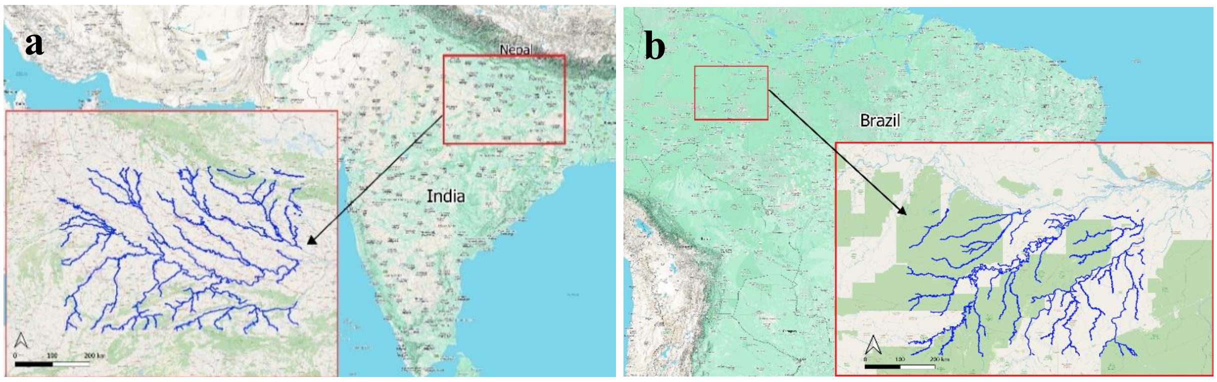


Figure 3: The two study areas, a) the Himalayas in Asia, and b) the Amazon in South America.

All the rivers are stored as separate "reaches" (section of a river with roughly uniform hydrological conditions) and all have a unique *reach_id*. The different reaches and centerlines in the study areas are clipped into a rectangle image using the maximum ends of the segment with a buffer of 10%, so that each segment of the river is in one image. The rectangle shape is chosen for its ease of standardization in the models, while the buffer serves to add supplementary context for labeling the rivers. The useless segments with a height or width of less than 30 pixels, or containing a grey area, such as lakes, are deleted. An example of a clipped river- and centerline is shown in Figure 4. The images may include tributaries but their presence is retained for the model to determine their relevance in classifying the river types. Figure 4 is a clear example of a fully braided river, although numerous segments have multiple river types in one image. Consequently, the images are labeled with fractional values that collectively sum to one, ensuring that each image can be labeled with different types if necessary. An example of a river where two types are apparent in one image is illustrated in Figure 5 on the next page. This river is classified as 90% braided and 10% anastomosing.

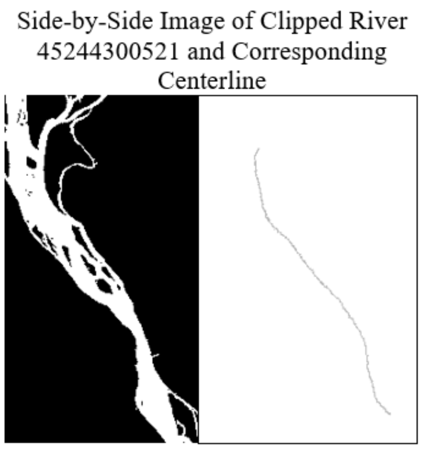

Side-by-Side Image of Clipped River 45244300521 and Corresponding Centerline

Figure 4: Example images of the clipped segment and the centerline of river 45244300521.

River 45243600611

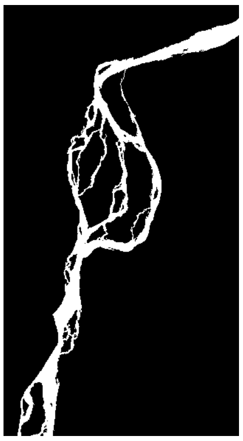


Figure 5: Example image of river 45243600611.

Each clipped segment of the river is manually labeled by two to four persons to ensure more reliability. Due to the study's novelty and limited time, the workflow is applied to a small subset of the global data.

As previously mentioned, the labeled dataset contains fractions of each river class and some have a certainty added how sure the person is about the labeling. For each image, the highest fraction is set to one and the rest to zero as a (D)CNN requires binary input. However, when a river segment is over 30% anastomosing and for the remainder meandering or wandering, the river label is set to anastomosing. The same applies to rivers classified as 30% braided and the rest meandering and/or wandering. This approach aims to increase the sample size for these labels anticipating that the model will have more difficulty with them due to the limited sample size. The other two river morphologies have more samples. The distribution of rivers per region is illustrated in Figure 6 and the total distribution of all the rivers is shown in Table 1. As can be seen in Figure 6, among the different classes in Asia, meandering is the most occurring and the other types have lower counts and wider spread. South America however predominantly has wandering rivers, with a notable scarcity of both anastomosing and braided rivers.

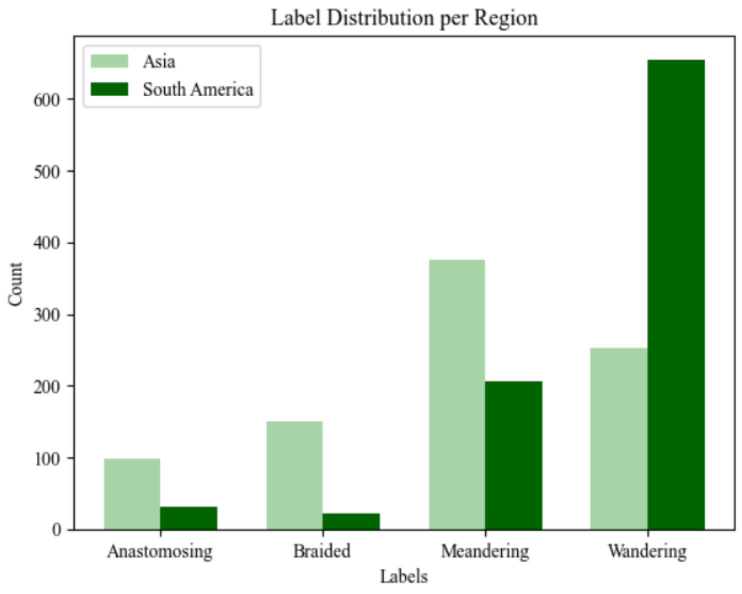


Figure 6: River morphology distribution per region.

Table 1: Distribution of the four river morphologies.

Anastomosing	**Braided**	**Meandering**	**Wandering**
131	172	582	907

3. Methodology

3.1 (D)CNN

(Deep) Convolutional Neural Networks, a form of deep learning, are primarily applied in the field of pattern recognition within visual imagery (Saxena, 2022). The conceptual inspiration is the brain, therefore the design of these artificial neural networks follows the function of the brain closely (Lindsay, 2021). They however follow a simplified process for computational efficiency. A (D)CNN for classification processes the input images through multiple layers, which perform various operations to produce the desired output; a classification label. The typical architecture of a (D)CNN is shown in Figure 7. It starts with feature extraction and finishes with classification. The convolutional layer is the most important operation of a (D)CNN. The convolutional layers apply filters that slide over the image to detect features such as edges or patterns (Yamashita et al., 2018). This process is called convolution and is an element-wise product of the filter and overlapping image sections. Figure 8 illustrates a convolution with a 3×3 filter and how a feature is calculated. The weights in the filter are initially random and are adjusted by the network based on backpropagation to optimize its ability to detect features (Lecun et al., 1998). After convolution, an activation function, often Rectified Linear Unit (ReLU), is utilized within the same layer to introduce non-linearity which helps the network to learn complex patterns. The output of the convolutional layer is a set of feature maps (Lindsay, 2021), and each feature map highlights different aspects of the input images. Subsequently, the pooling layer reduces the dimensionality of these feature maps to improve computational efficiency (Yamashita et al., 2018; Saxena, 2022). A (D)CNN can have a series of convolutional layers. The more convolutional layers, the more abstract the feature maps become since both the filter and the feature maps become a pattern within a pattern within a pattern. The last stage of the network is used for classification. The neural network requires a one-dimensional vector as input, which is achieved by applying the flatten layer. This layer stacks all the feature units together forming as input for the fully connected layer (Yamashita et al., 2018). The fully connected layer also called the dense layer, connects every unit to each possible classification with different learned weights (Gu et al., 2018; Saxena, 2022). Finally, the last layer employs a softmax operation that converts the output to probabilities that indicate the likelihood of the image belonging to each category (Yamashita et al., 2018). A CNN is typically considered as deep if it contains more than five layers (Mallat, 2016). In addition, Basha et al. (2020) mention a DCNN consists of convolution, pooling, and fully connected layers.

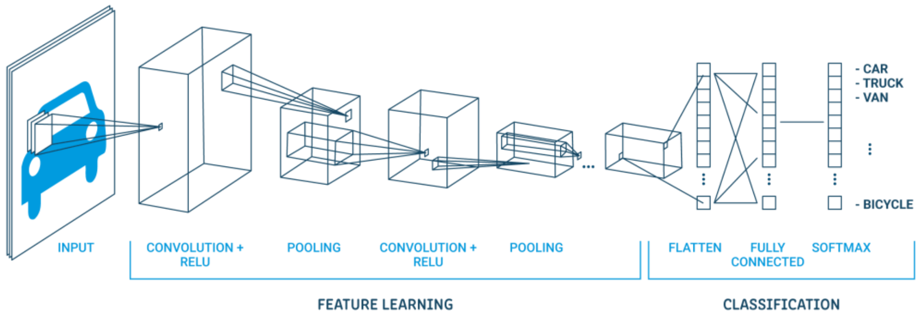


Figure 7: Example of a (D)CNN architecture (Harvey, 2024).

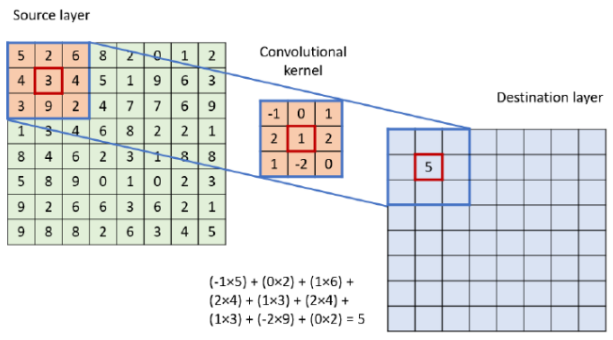


Figure 8: Schematic illustration of a convolution operation (Podareanu et al., 2019).

3.2 Prepare for training

To ensure uniformity in processing, a (D)CNN mandates a consistent input size for every image, which is standardized to 128 by 128 pixels. In general, the resolutions for (D)CNNs range between 64 by 64 pixels and 256 by 256 pixels (Thambawita et al., 2021). The chosen input size strikes a balance between resolution and computational overload. An example for each of the river morphologies is displayed in Figure 9.

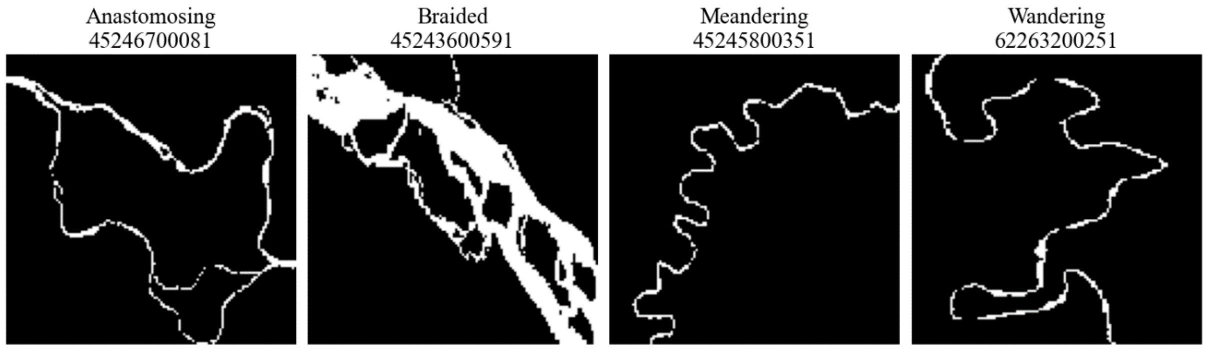


Figure 9: Examples of standardized images for each river morphology.

A total set of 1,792 images is divided into training, validation, and test sets using a 70:15:15 ratio, respectively. The distribution presented in Table 1 of Chapter 2.2, indicates a class imbalance. To prevent this, Synthetic Minority Oversampling Technique (SMOTE) Tomek, inspired by Tahir et al. (2023), was tested on the training data to generate new synthetic images. Reza and Ma (2018) describe how SMOTE generates synthetic samples of the minority class using three steps. They state SMOTE first chooses a random observation of the minority class and subsequently among its 5 nearest minority class neighbors, an instance is selected. Finally, a new sample is created by randomly interpolation the two samples (Reza & Ma, 2018). Tomek links are used to eliminate overlapping instances by removing instances that are considered to belong to multiple classes caused by SMOTE (Sasada et al., 2020). However, this produces many unusable images so this technique is not used in the final method. Examples of unusable images can be found in Figure A1 in Appendix A on page 38. The chosen method to fix the class imbalance issue in the training data is Random Oversampling (ROS). ROS randomly duplicates minority class examples until the classes are balanced, where every class less than the largest is considered a minority class (Johnson & Khoshgoftaar, 2019). In this case, ROS upscales the samples in each class to the majority class (wandering) resulting in equal classes of 619 in the training set. While this technique aims to mitigate class imbalance, it can potentially result in overfitting due to the duplicate images in the training set.
This study evaluates several (D)CNN architectures including single and dual image input networks. To ensure reliable comparison, identical training, validation, and test splits, along with consistent resampling (ROS), is employed for both input datasets.

3.3 Training and validation process

To determine the most suitable (D)CNN architecture for this classification task, several experiments are employed. These experiments are designed to create one model with a single input comprising images of the clipped rivers, and another model with two inputs that include images of both clipped rivers and centerline images from the SWORD dataset. The centerline could give additional information to the model to exclude patterns of the tributaries. The (D)CNN architecture for the two-input models contains two branches with their own sets of layers. The branches include the layers specified in the upcoming experiments. The two branches are concatenated followed by the fully connected layer and the output layer. The basic architecture for the two-input (D)CNN is shown in Figure 10 on page 8.

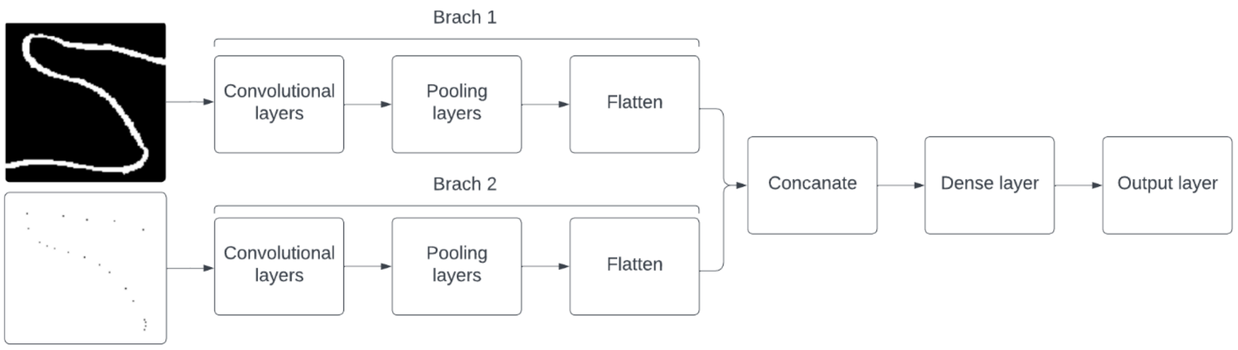


Figure 10: Architecture of a two-input (Deep) Convolutional Neural Network using the image and centerline.

Experiments 1 to 5 are trained for 30 epochs with a batch size of 32, and an early stopping with a patience of 10 is implemented to prevent overfitting. This regularization technique can obtain a model with a better validation set error by returning the parameter setting at the point in time with the lowest validation error (Goodfellow et al., 2016). A learning rate schedular of 0.001, decreasing by a factor of 0.99 after each epoch is implemented to improve stability during training. The optimizer utilized for all models is Adam.

Typically when a (D)CNN has a small dataset, transfer learning is applied, but using the VGG-19 architecture did not yield satisfactory results, therefore, new models are created to explore their performances.

3.3.1 Experiment 1

The first experiment is finding the number of convolutional layers. Five pairs have been considered, each following a pooling layer. Using six pairs would reduce the image excessively. As mentioned before, the input size is 128 × 128 pixels. After one pair, the shape becomes 64 × 64 pixels; after two pairs 32 × 32 pixels, and so forth, until after five pairs the image is reduced to 4 × 4. The number of feature maps for the layers are 24, 48, 64, 128, and 256 each with a kernel (convolutional filter) size of 5 × 5 and a dense layer with 156 units. The parameter padding is to "same" to maintain the spatial dimensions of the input and prevent information loss.

3.3.2 Experiment 2

This experiment determines the number of feature maps. The second experiment is executed applying the best outcome of experiment 1. Various feature maps, using multiple networks, are employed for each convolutional layer in the network. The formula used for the number of feature maps in each convolutional layer in this experiment is defined in Equation 1. This formula ensures a geometric increase of feature maps for each layer, starting from eight feature maps in the initial convolutional layer of the first network. The number of filters grows when the depth of a (D)CNN increases to enable the model to learn and capture more complex patterns.

$$F_{j,k} = \begin{cases} 8 & \text{If } j = 0 \text{ and } k = 0 \\ (j+1) \times 2^{k+3} & \text{otherwise} \end{cases} \qquad (1)$$

Where:
j: the index of the network $j \in \{0, 1, 2, \ldots\}$,
k: the index of the convolutional layer $k \in \{0, 1, 2, \ldots\}$.

When utilizing this technique, it is essential to guarantee the networks do not become overly complex due to an abundance of feature maps. Given an example of an output of five convolutional layers in experiment 1, the maximum of four networks will be considered in experiment 2. The highest count of feature maps results therefore in 512. When the number of convolutional layers decreases, the number of networks that can be considered increases.

3.3.3 Experiment 3

The third experiment focuses on the amount of dense units in the fully connected layer. These units can affect the models' capacity, complexity, and performance. A higher number of units allows the models' capacity to learn more complex relationships in the images leading to a better performance on the training set. However, increasing the amount of units also increases the necessary memory and computational power to train and evaluate the model. Apart from that, it might also lead to an overfitted model due to the complex patterns and learning to memorize the training data rather than the test data. Conversely, using an insufficient amount of units may result in underfitting where the model fails to capture important patterns and has poor performances. (Chen 2021; Basha et al., 2020; Berngardt 2024) This experiment used the optimal model structure of experiment 2 and Equation 2 shows the formula used to determine the number of dense units. The maximum of 512 units ensures the model doesn't become too complex.

$$D_j = \begin{cases} 0 & \text{If } j = 0 \\ 2^{j+4} & \text{otherwise} \end{cases} \qquad (2)$$

Where:
j: the index of the network $j \in \{0, 1, 2, 3, 4, 5\}$

3.3.4 Experiment 4

To prevent overfitting, different dropout rates after each pooling layer, and between the fully connected and the output layer have been analyzed for the model with the best outcome in experiment 3. Dropout is a technique where randomly selected neurons (processing units within the layers) are ignored (dropped out) during the training process which encourages the network to learn more robust features (Cai et al., 2019). Applying too much dropout can lead to an underfitted model. The rates 0%, 10%, 20%, 30%, 40%, 50%, 60%, and 70% are placed as previously mentioned.

3.3.5 Experiment 5

The last experiment is to explore more advanced features. The optimal model aggregated from experiment 4 is utilized to explore the performance of three advanced features, namely weight decay, batch normalization, and data augmentation. Weight decay can reduce the complexity of the (D)CNN by restricting the value of the neuron, thereby reducing the overfitting to a certain extent (G. Li et al., 2022). A reasonable value ranges between 0 and 0.1 (Kuhn & Johnson, 2013). The L2 regularization term added to the overall loss function during training is formulated in Equation 3.

$$\lambda \sum_{i=1}^{L} ||W_i||_2^2 \qquad (3)$$

Where:
λ: the regularization parameter (weight decay of 0.0001),
L: total number of layers,
W_i: weight matrix of the i-th layer,
$||W_i||_2^2$: the squared Frobenius norm of W_i, which is the sum of the squares of all elements in W_i.

Batch normalization can lead to faster convergence during training which improves stability, and also helps to prevent overfitting (Bjorck et al., 2018). This is achieved by normalizing the activations of each layer. Batch normalization interacts with the learning rate and can make the learning rate schedule more effective since normalization ensures that even significant changes in the learning rate do not destabilize training (Ioffe & Szegedy, 2015). A model without data augmentation processes the same images per epoch but a model with data augmentation processes augmented versions of the images per epoch to provide more diverse data. It has been shown that data augmentation improves the generalization of training (Rebuffi et al., 2021).
The first model has a weight decay, L2 regularization of 0.0001, implemented in the convolutional layers to control the magnitude of the weights. The second model contains batch normalization layers

between the convolutional and pooling layers. The third model has batch normalization layers, and data augmentation is used to analyze whether it increases performance. The parameters of the employed data augmentation are shown in Appendix A, Table A1 on page 38. An example of an image with augmentation is shown in Figure 11. Finally, another model is created that contains all three advanced features.

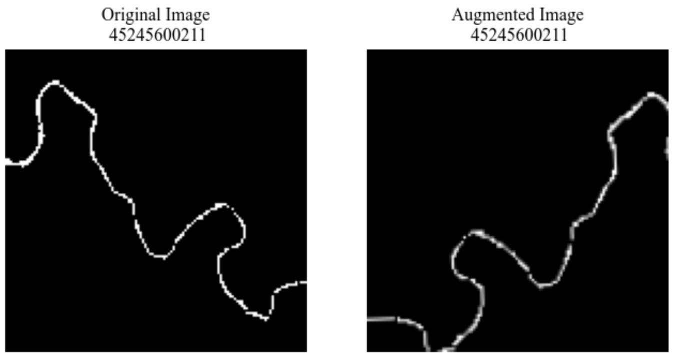


Figure 11: Original and augmented image of river 45245600211.

3.3.6 Training metrics

The metrics used to evaluate the model during training and validation are accuracy (Equation 4) and the categorical cross-entropy loss, Equation 5. The accuracy reflects on how well the model performed and is a percentage of correctly classified samples out of the total samples in a range of [0,1]. The loss measures how well the model's classified probability distribution matches the actual distribution of class labels and ranges from [0, INF+). When weight decay is applied as L2 regularization in the last experiment, the loss has an additional term (Equation 3), resulting in Equation 6, that penalizes the large weights. The goal is to minimize the loss and to maximize the accuracy of the model. A high loss does not necessarily mean a low accuracy, but the loss penalizes confident incorrect predictions heavily which can inflate the loss even if the accuracy is high (Qui˜nonero-Candela et al., 2006).

$$Accuracy = \frac{\#\, correct\ classifications}{total\ \#\, classifications} \times 100\% \tag{4}$$

$$Loss_{total} = -\frac{1}{N}\sum_{j}^{N}\sum_{i}^{4} y_{ij}\log(\hat{y}_{ij}) \tag{5}$$

$$Loss_{reg} = -\frac{1}{N}\sum_{j}^{N}\sum_{i}^{4} y_{ij}\log(\hat{y}_{ij}) + \lambda\sum_{k=1}^{L}||W_k||_2^2 \tag{6}$$

Where:
N: number of samples in the dataset,
y_{ij}: true label for i-th class of the j-th sample,
$\hat{y}_{ij}$: the predicted probability over the i-th class of the j-th sample,
λ: the regularization parameter (weight decay of 0.0001),
L: total number of layers,
W_k: weight matrix of the k-th layer,
$||W_k||_2^2$: the squared Frobenius norm of W_k, which is the sum of the squares of all elements in W_k.

3.4 Testing process

The results are analyzed using both the weighted F1-score and the F1-score to check the models' performances on unseen data. The models' classifications can be visualized using the confusion matrix, illustrated in Table 2. The values in the confusion matrix represent the number of predictions of each class.

Table 2: Confusion matrix of the four class classifications.

		Predicted			
		Anastomosing	**Braided**	**Meandering**	**Wandering**
Actual	Anastomosing	x_{11}	x_{12}	x_{13}	x_{14}
	Braided	x_{21}	x_{22}	x_{23}	x_{24}
	Meandering	x_{31}	x_{32}	x_{33}	x_{34}
	Wandering	x_{41}	x_{42}	x_{43}	x_{44}

From the confusion matrix, the precision, recall, and F1-score can be derived. The F1-score is used to evaluate the models' performances for each class on the test set since the accuracy is biased due to class imbalance. The F1-score is a more balanced evaluation since it is the harmonic mean of the precision (Equation 9) and the recall (Equation 10) (Dalianis, 2018). Since there is a class imbalance, the overall model's performance is evaluated using the weighted F1-score. The weighted F1-score considers the class weight in the test data. Equations 7 to 13 show how the weighted F1-score and the F1-score per class are calculated.

$$\text{Weighted F1} - \text{score} = \sum_{i=1}^{4} w_i \times F1 - score_i \tag{7}$$

Where
w_i: weight of class i, proportion of the true instances of class i in test dataset,

$$\text{F1} - \text{score}_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i}, \tag{8}$$

Where

$$Precision_i = \frac{TP_i}{TP_i + FP_i}, \tag{9}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i}. \tag{10}$$

Where

$$TP_i = x_{ii} \tag{11}$$

$$FP_i = \sum_{j=1, j\neq i}^{4} x_{ji}, \tag{12}$$

$$FN_i = \sum_{j=1, j\neq i}^{4} x_{ij}. \tag{13}$$

3.5 Baseline performance

To establish a baseline for evaluating the performance of the (D)CNN that is created, the scenario of majority class guessing is assessed due to class imbalance. The baseline weighted F1-score of testing the (D)CNN is 41%. This score is predominantly influenced by the wandering class, which achieves an F1-score of 73%, while all other classes achieve a score of zero in this case. Proof for all F1-scores based on the majority class baseline can be found in Appendix A page 38. Achieving higher scores than the baseline demonstrates the model's ability to learn patterns in the river images and make better predictions beyond what would be expected by predicting the majority class for each image. To compare the performances, Cohen's d is measured to quantify and compare improvements of the model relative to the baseline method. Cohen's d measures how large an effect is (Baguley, 2009). A $d = 0.2$ indicates a small effect, $d = 0.5$ a medium effect, and $d = 0.8$ a large effect (Cohen, 2013). Equations 14 and 15 show how the effect sizes are calculated for the overall performance and for each class.

$$d_{overall} = \frac{\mu_{DCNN} - \mu_{baseline}}{\sqrt{\frac{\sigma_{DCNN}^2 + \sigma_{baseline}^2}{2}}} \tag{14}$$

$$d_i = \frac{F1_{DCNN}^{(i)} - F1_{baseline}^{(i)}}{\sqrt{\frac{\sigma_{DCNN}^2 + \sigma_{baseline}^2}{2}}} \tag{15}$$

Where
μ: means of F1-scores,
σ: (pooled) standard deviations of the F1-scores,
$F1^{(i)}$: F1-score class i.

4. Results and analysis

This chapter presents the results of the five experiments utilized to find the most suitable (D)CNN architecture for both single input and dual inputs. Next, the best model is selected to conduct predictions on the test dataset.

4.1 Experiments

4.1.1 Convolutional layers

One-input model

The losses and accuracies of the one-input models that have been created for the first experiment are shown in Table 3, where ‘C-P’ denotes a pair of convolutional and pooling layers. All different depths have a consistently, exceptionally high training accuracy and low training loss, indicating that all models accurately fit the training images. Figure 12 shows the training loss of all models converges rapidly to nearly zero. (C-P) ×3, ×4, and ×5 show a more gradual decrease in training loss, (C-P) ×5 being the slowest, compared to model (C-P) ×1. The model with one pair of convolutional and pooling layers remains low over epochs. Networks with more convolutional and pooling layers tend to have lower validation losses. The validation loss increases over epochs indicating the models are overfitting, as shown in Figure 13 on page 14, and is also higher compared to the training loss. The validation accuracies are significantly lower compared to the training accuracy also indicating all models show signs of overfitting. The one-input model with five convolutional and pooling layers is chosen to conduct experiment 2 since it has the lowest validation loss and the highest validation accuracy, indicating better generalization performance on unseen data.

Table 3: Performances of the five one-input models in experiment 1.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN (C-P) ×1	0.003	2.587	1.000	0.494
CNN (C-P) ×2	0.101	1.829	1.000	0.472
CNN (C-P) ×3	0.275	1.818	1.000	0.483
CNN (C-P) ×4	0.338	1.510	1.000	0.520
CNN (C-P) ×5	0.769	1.175	0.998	0.535

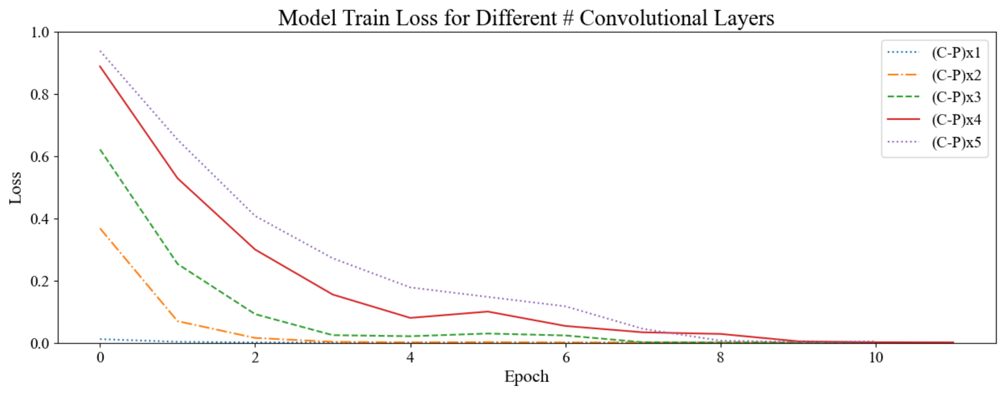


Figure 12: Model train loss over epochs of the five one-input models of experiment 1.

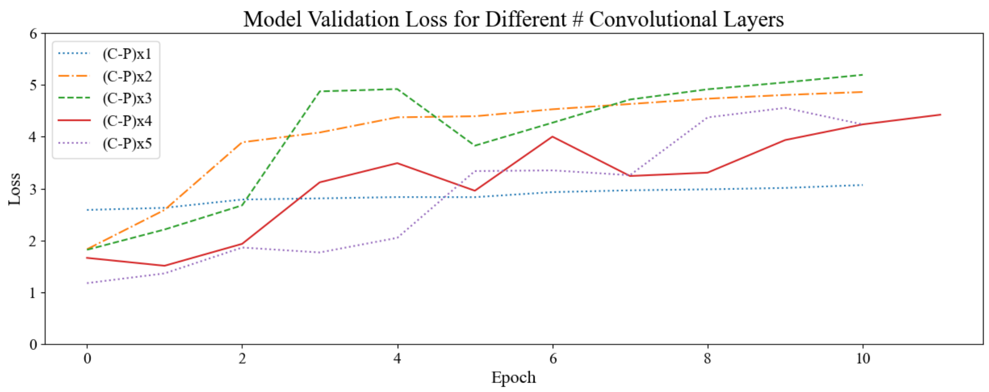


Figure 13: Model validation loss over epochs of the five one-input models of experiment 1.

Two-input model

The performances of experiment 1 using two inputs are shown in Table 4. All networks also have a high training accuracy. The training loss increases when more pairs of convolutional and pooling layers are utilized whereas the validation loss decreases. Increasing the depth of the two-input networks has an effect on the performance on unseen data. The additional Figure 14, shows the training loss of all models converges to almost zero with the model (C-P) ×1 the quickest and (C-P) ×5 being the slowest. Figure 15 on the next page shows all models are overfitting when the epochs increase. The network with five pairs of layers is the second least overfit, following the network with only one pair. Given the highest validation accuracy and the lowest validation loss, five pairs of convolutional and pooling layers are employed for further analysis of the two-input CNN.

Table 4: Performances of the five two-input models in experiment 1.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN (C-P) ×1	0.176	1.336	1.000	0.491
CNN (C-P) ×2	0.508	1.387	1.000	0.513
CNN (C-P) ×3	0.932	1.250	0.999	0.450
CNN (C-P) ×4	1.118	1.180	1.000	0.472
CNN (C-P) ×5	1.117	1.158	0.994	0.520

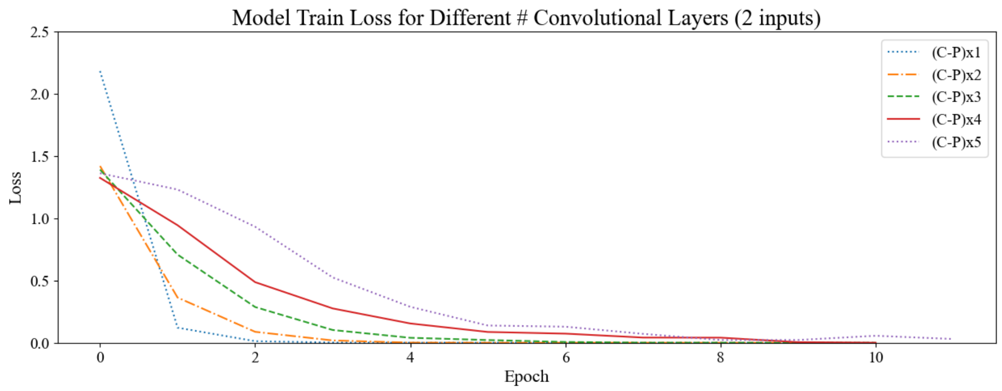


Figure 14: Model train loss over epochs of the five two-input models of experiment 1.

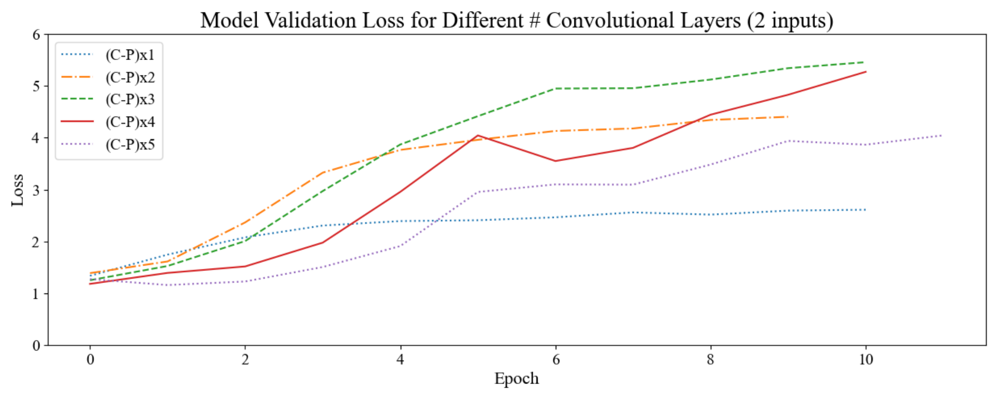


Figure 15: Model validation loss over epochs of the five two-input models of experiment 1.

Experiment 1 demonstrates that both single and dual input models perform better using deeper networks to classify unseen data. Both networks have five convolutional layers, five pooling layers, a flatten layer, and a fully connected layer. This shows the necessity of deeper architectures for classifying river segments.

4.1.2 Feature maps

One-input model

Given the DCNN with five convolutional and pooling layers, this experiment tested four networks, as explained in Chapter 3.2.1. The performances of these four employed models with a different number of feature maps in the convolutional layers are displayed in Table 5. As the number of feature maps increases, the training accuracy remains consistently high across all models; however, there is no corresponding improvement visible in the training loss. The validation loss shows variability but no clear in- or decreasing trend. The network with 24 maps in the first layer achieves the lowest validation loss. The validation accuracy improves as the network widens. All models converge fast (Figure 16) and show signs of overfitting (Figure 17, page 16). This experiment shows complexity of feature maps likely helps the network learn more intricate patterns in the data. The network with 32 feature maps in the first layer exhibits one of the lowest validation losses and the highest validation accuracy making it the most preferable out of these four, and is therefore chosen to employ experiment 3. The final feature maps in the five convolutional layers are 32, 64, 128, 256, and 512 for the network with one input.

Table 5: Performances of the four one-input models in experiment 2.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN 8 maps in first layer	1.077	1.213	0.999	0.491
CNN 16 maps in first layer	1.260	1.301	0.995	0.480
CNN 24 maps in first layer	1.092	1.146	0.997	0.487
CNN 32 maps in first layer	1.303	1.200	0.988	0.528

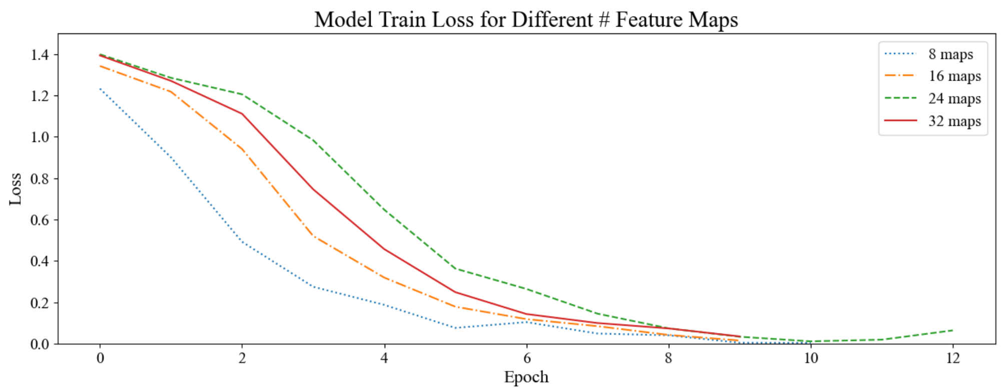


Figure 16: Model train loss over epochs of the four one-input models of experiment 2.

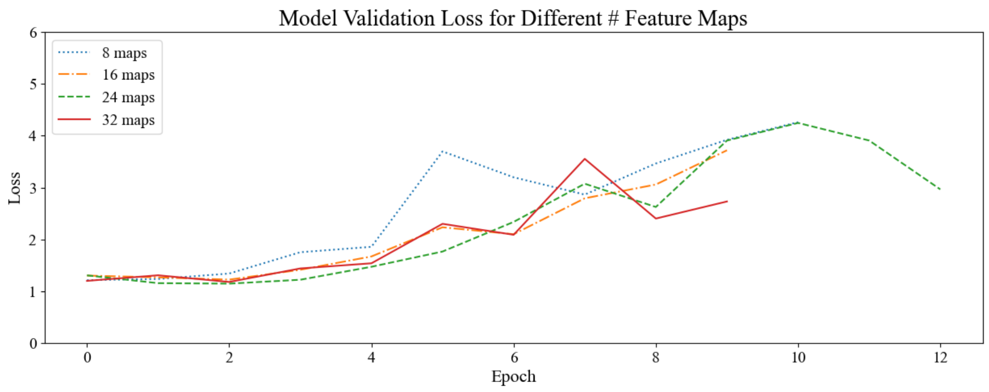


Figure 17: Model validation loss over epochs of the four one-input models of experiment 2.

Two-input model

The performance of the networks that used two inputs is shown in Table 6. Since these networks also have five convolutional and pooling layers, the same feature map options are considered as for the one-input model. All the networks have learned the training data well and converged quickly, as shown in Figure 18. The model with the highest number of feature maps in the first layers converges slower than the rest due to more complexity. An increase in the number of feature maps per layer leads to a reduction in training loss. The training loss shows a significant drop when the network employs 16 feature maps in the first layer compared to 8 maps. All models are overfitting when the epochs increase as can be seen in Figure 19 on page 17. The validation losses show a decrease when the number of feature maps increases. The network with 24 maps in the first layer has the highest validation accuracy and one of the lowest validation losses and is applied to experiment 4. The number of feature maps in each convolutional layer is 24, 48, 96, 192, and 384.

Table 6: Performances of the four two-input models in experiment 2.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN 8 maps in first layer	1.259	1.408	0.994	0.494
CNN 16 maps in first layer	0.888	1.137	0.992	0.494
CNN 24 maps in first layer	0.750	1.113	0.996	0.524
CNN 32 maps in first layer	0.764	1.083	0.999	0.498

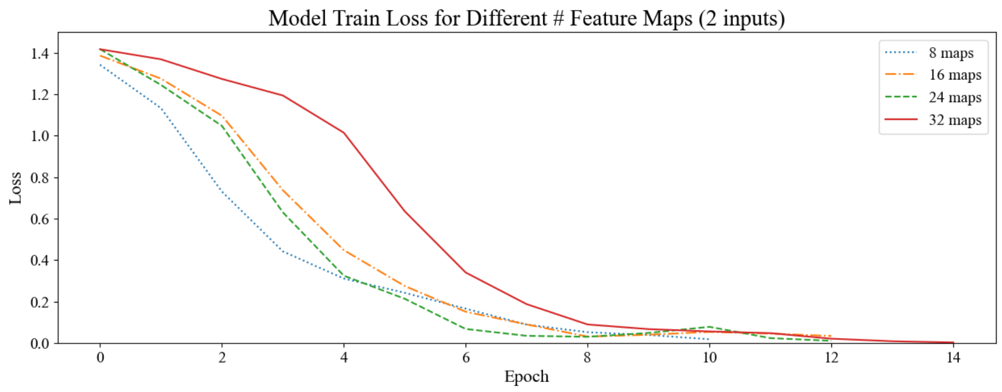


Figure 18: Model train loss over epochs of the four two-input models of experiment 2.

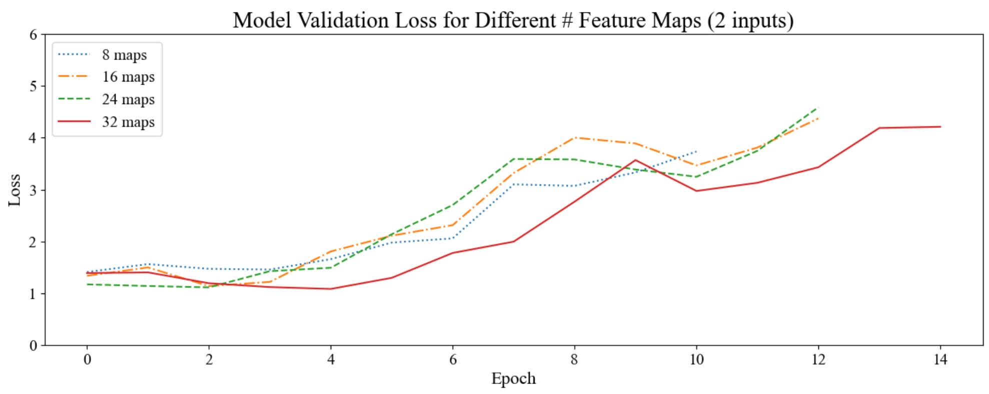


Figure 19: Model validation loss over epochs of the four two-input models of experiment 2.

Experiment 2 reveals that a network with one input benefits from a wider (more feature maps) network, while the two-input DCNN performs better with a slightly narrower network. The one-input model with an increased number of feature maps allows for more detailed feature extraction, capturing more patterns from one input image. In contrast, the two-input model leverages two input images, reducing the need for extremely wide feature maps since it relies on combined information from the river image and the centerline to achieve effective feature extraction.

4.1.3 Dense units

One-input model
Table 7 shows the performances of the one-input networks for experiment 3. All models except the network without dense units and 128 units classify excellent on the training data according to the training accuracy. The model without a fully connected layer cannot sufficiently learn to capture the patterns effectively within the river images. Increasing the number of units in the fully connected layer shows varied results, indicating that the number of dense units affects the performance on the training data. Remarkably, the network with 128 units has a constant training loss and does not converge, as can be seen in Figure 20 on page 18, suggesting the training process has issues. The model does not learn the train data well since it has a low train accuracy. The network also shows unstable accuracy compared to the others (Figure B5 of Appendix B on page 41), confirming the issues during training. This could be due to poor initialization of the weights. If weights are initialized too small or large, it can impact the learning process. The validation loss increases with a denser fully connected layer up to 256 units. The model with 512 dense units provides the best validation loss and accuracy for this particular architecture indicating it strikes a fine balance between complexity and generalization. The overfitting issue is still evident during this experiment as can be seen in Figure 21 on the next page and in the gap between training and validation accuracy in Table 7. The requirement of 512 units in the fully connected layer indicates that the model needs more complexity to capture the patterns in the images and gives sufficient capacity to learn the features present in the data. Experiment 4 is therefore applied with 512 units in the fully connected layer.

Table 7: Performances of the six one-input models in experiment 3.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN 0 dense units	1.002	1.203	0.100	0.513
CNN 32 dense units	0.721	1.203	0.999	0.498
CNN 64 dense units	1.386	1.382	0.976	0.480
CNN 128 dense units	1.386	1.385	0.250	0.327
CNN 256 dense units	1.315	1.442	0.993	0.509
CNN 512 dense units	1.125	1.060	0.993	0.539

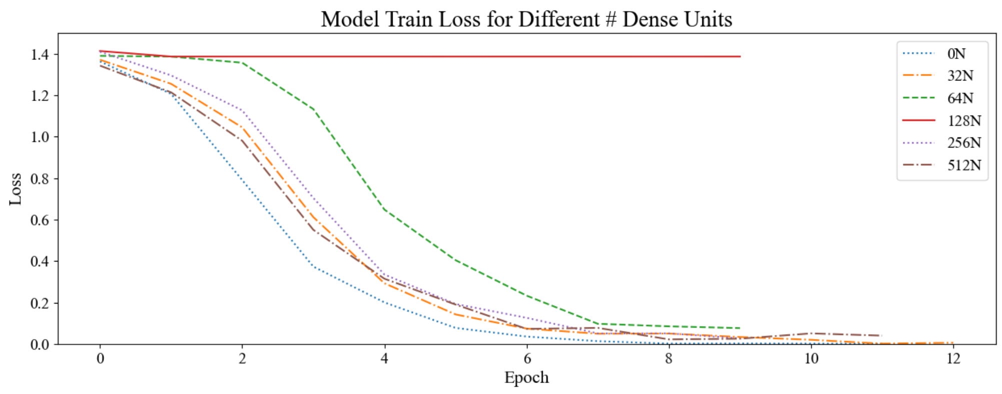


Figure 20: Model train loss over epochs of the six one-input models of experiment 3.

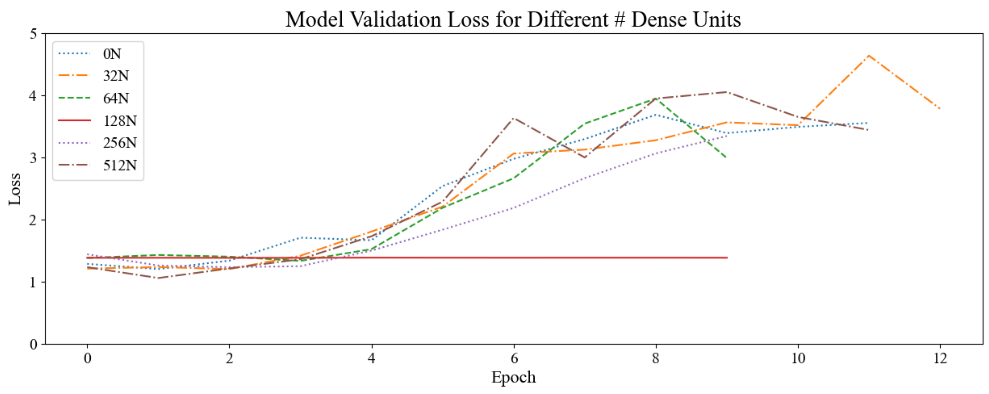


Figure 21: Model validation loss over epochs of the six one-input models of experiment 3.

Two-input model

The corresponding performances of the six two-input networks, conducted in experiment 3, are shown in Table 8 on page 19. The networks with 256 and 512 dense units in the fully connected layer are the ones that perform poorly on unseen data, exhibiting a low train accuracy and the highest train losses compared to the other networks. The low train accuracy and high validation accuracy of 256 units suggest the model is underfitted. There is no balance between the capacity to learn complex relationships (size of dense layers) and the overall complexity of the model. The associated accuracies fluctuate over epochs which is shown in Figure B6 of Appendix B on page 41. The network with 512 dense units reveals issues during training as shown in Figure 22 on the next page, whereas the other networks converge to zero. Remarkably, the model without a fully connected layer performs well overall. It demonstrates that complex tasks can be tackled effectively through hierarchical feature extraction and without the additional complexity of dense connections. The validation loss decreases steadily as the number of units in the fully connected layers increases up to 64 units. However, it spikes at 128 units before decreasing again and then increases notably for the last three models. The training loss and accuracy also show varying patterns across different configurations. The validation accuracy does not show clear improvement or deterioration with a denser network. During this experiment, overfitting issues still occur for the trained models (Figure 23 page 19). The most optimal number of dense units in the fully connected layer in the two-input DCNN is 64 units, having the highest validation accuracy and lowest validation loss.

Table 8: Performances of the six two-input models in experiment 3.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN 0 dense units	0.882	1.127	0.998	0.517
CNN 32 dense units	1.161	1.112	0.972	0.491
CNN 64 dense units	1.253	1.110	0.990	0.539
CNN 128 dense units	1.280	1.934	0.994	0.509
CNN 256 dense units	1.386	1.384	0.252	0.502
CNN 512 dense units	1.386	1.381	0.265	0.327

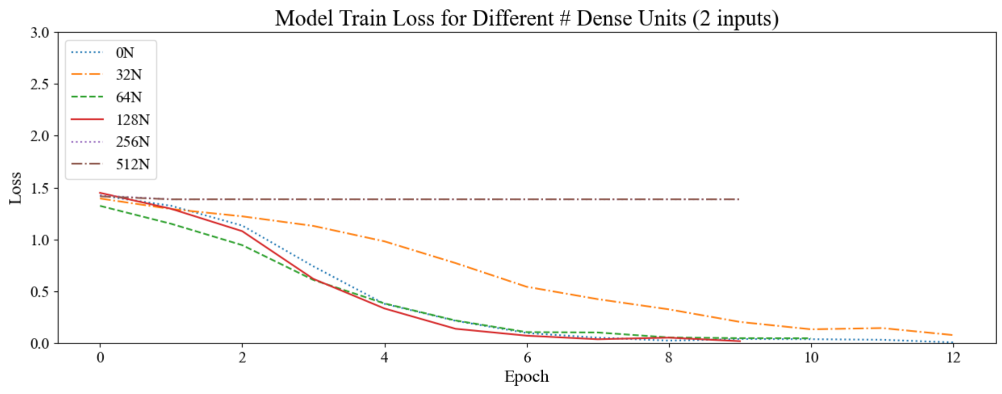


Figure 22: Model train loss over epochs of the six two-input models of experiment 3.

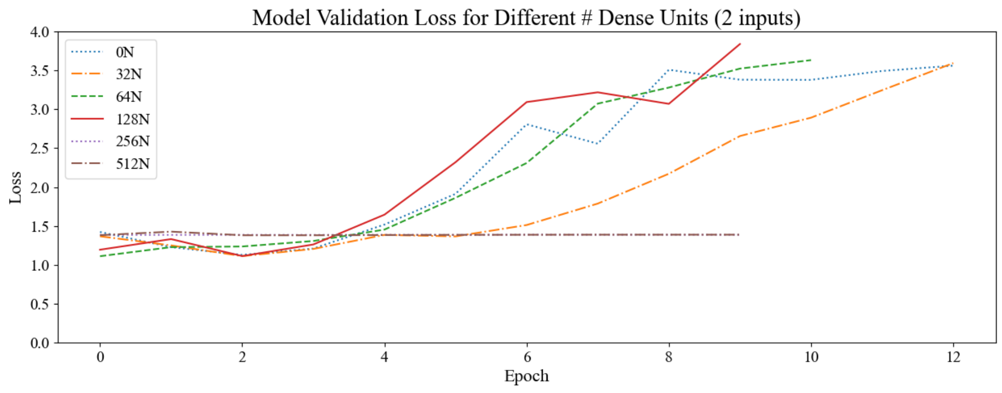


Figure 23: Model validation loss over epochs of the six two-input models of experiment 3.

The optimal number of dense units varies for the one- and two-input models: the one-input model achieves its best performance with 512 units, whereas the two-input model performs the most optimal with 64 units. The difference can be attributed to the need to handle more complexity (more feature maps) and requiring a larger capacity in the dense layer while the two-input model benefits from fewer parameters for effectiveness. Experiment 3 shows the one-input DCNN gains more from a highly connected network in contrast to the two-input DCNN. The chosen models for both single and dual input show signs of overfitting and might benefit from stronger regularization to avoid it.

4.1.4 Dropout rate

One-input model

The performances of the one-input networks with different dropout rates are shown in Table 9 on the next page. According to the training accuracy, increasing the dropout rate leads to underfitting. For dropout rates 0.3 to 0.7, the networks show signs of over-regularization, meaning the regularizations are too strong and the model becomes too simplistic to capture the patterns within the data. Too much dropout is not beneficial and hinders the network's ability to learn the patterns effectively, which results in the remainder of high losses, as shown in Figure 24. The train losses for the higher dropout rates stay approximately the same. The validation accuracy remains stable using higher dropout rates, but when using a dropout of 0.7, the validation accuracy drops significantly. In Figure 25 the effects of applying a dropout are evident as the validations losses remain relatively stable and do exhibit as significant increases over the epochs compared to earlier experiments. The validation loss however does still not converge to nearly zero with the use of dropout. Since higher dropouts do not improve the model's performance, a dropout rate of 0.1 is chosen to perform the last experiment despite the higher validation loss.

Table 9: Performances of the eight one-input models in experiment 4.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN D = 0	1.312	1.012	0.994	0.529
CNN D = 0.1	1.386	1.383	0.943	0.535
CNN D = 0.2	1.261	1.213	0.960	0.513
CNN D = 0.3	1.386	1.379	0.250	0.502
CNN D = 0.4	1.388	1.339	0.295	0.502
CNN D = 0.5	1.386	1.383	0.471	0.502
CNN D = 0.6	1.386	1.388	0.254	0.502
CNN D = 0.7	1.386	1.387	0.345	0.089

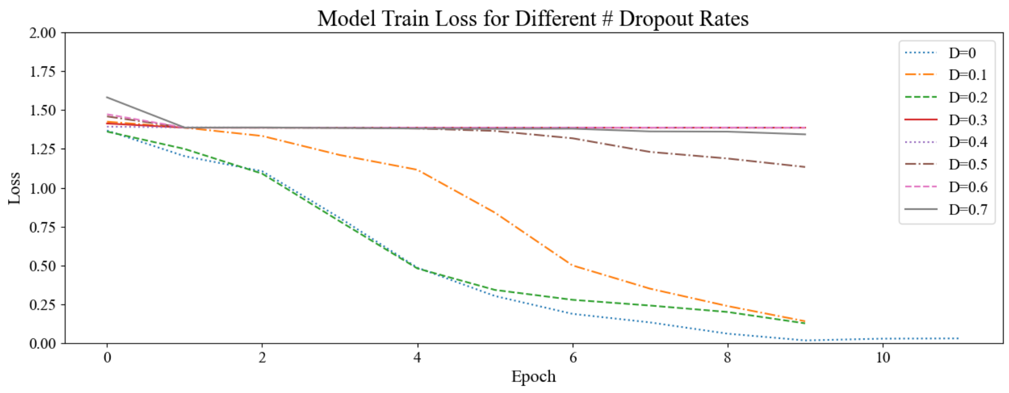


Figure 24: Model train loss over epochs of the eight one-input models of experiment 4.

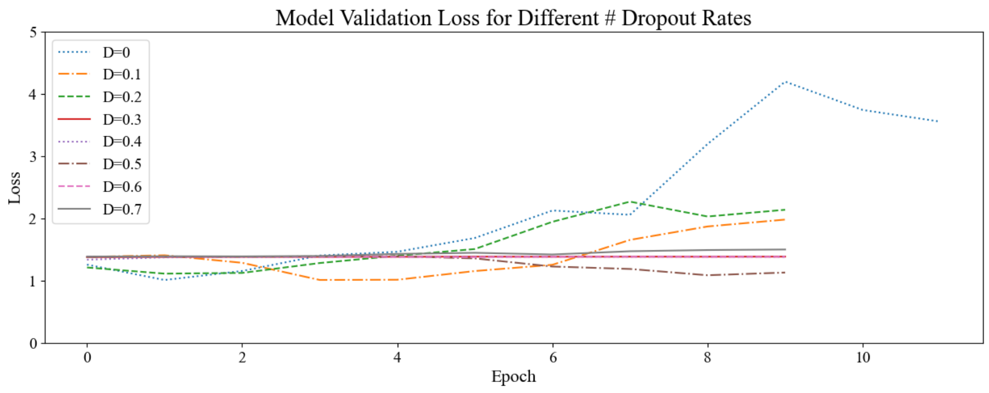


Figure 25: Model validation loss over epochs of the eight one-input models of experiment 4.

Two-input model

Table 10 on the next page shows the performances of the two-input models applied in experiment 4. Same as for the one-input models, the increase in dropout leads to underfitting. The train and validation loss is high for networks with a dropout rate of 0.3 to 0.7. The models fail to capture the patterns in the training images which is evidenced by the low train accuracy and high train loss. This was to be expected because dropout introduces randomness during training, preventing the network from relying too much on specific neurons. Beyond a dropout rate of 0.2, the validation accuracy starts to decrease and the validation loss increases, indicating large dropout rates hinder the ability to learn. They therefore do not converge either, as apparent in Figure 26. The dropout does affect the overfitting issue. In Figure 27 the validation loss does not increase as much as before. A dropout rate of 0.2 is utilized for the two-input DCNN since it has the highest validation accuracy and lowest validation loss.

Table 10: Performances of the eight two-input models in experiment 4.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN D = 0	1.160	1.253	0.999	0.517
CNN D = 0.1	0.974	1.183	0.972	0.550
CNN D = 0.2	0.883	0.994	0.933	0.558
CNN D = 0.3	1.359	1.392	0.781	0.494
CNN D = 0.4	1.382	1.374	0.536	0.439
CNN D = 0.5	1.386	1.375	0.443	0.509
CNN D = 0.6	1.386	1.388	0.345	0.506
CNN D = 0.7	1.386	1.382	0.250	0.502

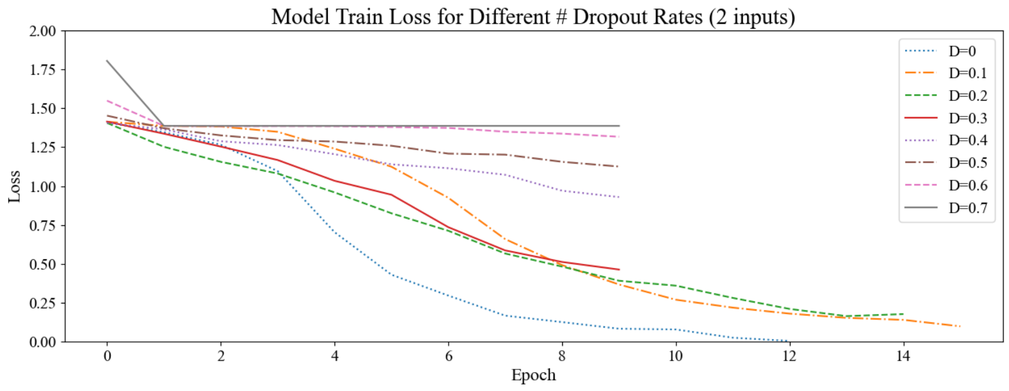


Figure 26: Model train loss over epochs of the eight two-input models of experiment 4.

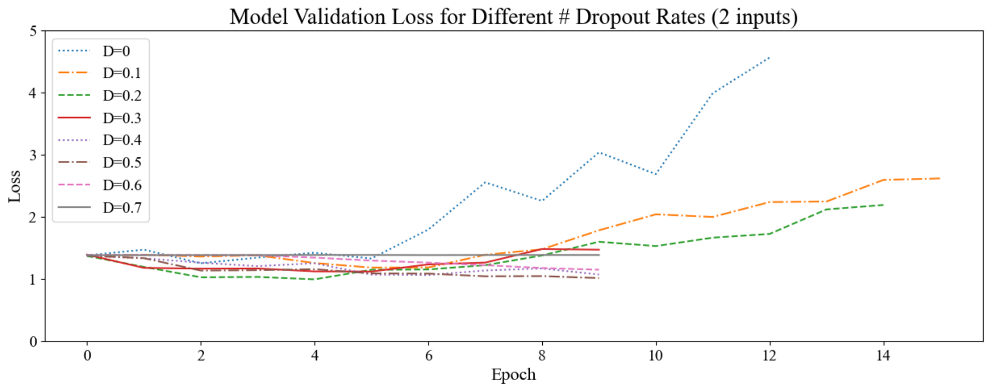


Figure 27: Model validation loss over epochs of the eight two-input models of experiment 4.

Experiment 4 shows a dropout rate of 0.1 is the most optimal for the single input DCNN and 0.2 for the dual input DCNN. The networks have varying complexities and therefore the need for appropriate regularization to optimize the performance is different. Even though the two-input model is less complex in terms of width, depth, and density, it benefits from a slightly higher dropout rate. Both

DCNNs are still slightly overfitted and the validation losses do not converge. Additional features were implemented in the next experiment to help improve this.

4.1.5 Advanced features

One-input model

The last experiment focused on advanced features, namely adding weight decay (WD), batch normalization (BN), data augmentation (DA), and combining all three. The corresponding results are shown in Table 11. The DCNN without any advanced features is still overfitted. Applying weight decay does not improve the train and validation losses or the train and validation accuracies, so the regularization has a negative effect on the performance. The model cannot capture the patterns in the training data and the loss remains the same over epochs, as shown in Figure 28. The accuracy also fluctuates significantly over each epoch illustrated in Figure B9 of Appendix B on page 43. The utilized weight decay is too strong for this network. Adding batch normalization to the network shows improvement in all metrics, indicating enhanced generalization and stability during training. Combining all three regularization techniques has the poorest performance among the employed models, indicating the regularization might be too aggressive or unsuited for this specific architecture and dataset. The networks that contain batch normalization alone, and batch normalization, data augmentation, and weight decay show convergence of validation accuracy, displayed in Figure 29 on the next page. The training loss of the last two models has not yet converged to nearly zero due to early stopping, showing a balance between under and overfitting. The model, however, is not entirely free from overfitting. Among all models, implementing batch normalization leads to the highest validation accuracy and the lowest validation loss. This results in the best generalization and is therefore the best-performing model for the single-input DCNN.

Table 11: Performances of the five one-input models in experiment 5.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN basic	1.386	1.383	0.943	0.535
CNN WD	1.390	1.390	0.268	0.502
CNN BN	0.388	1.299	0.971	0.546
CNN BN + DA	1.574	1.556	0.625	0.498
CNN BN + DA + WD	4.290	4.679	0.531	0.420

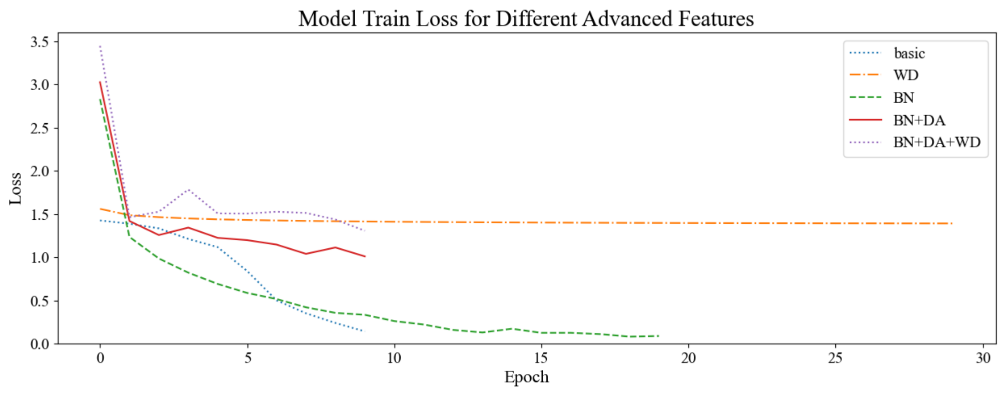


Figure 28: Model train loss over epochs of the five one-input models of experiment 5.

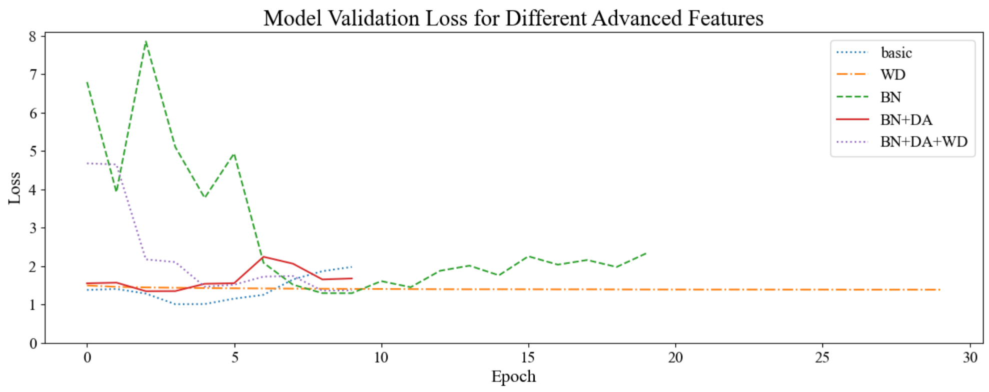


Figure 29: Model validation loss over epochs of the five one-input models of experiment 5.

Two-input model

Lastly the performances of the advanced features applied to the dual input DCNN are shown in Table 12. The training loss decreases by implementing weight decay and batch normalization separately but increases the validation loss. Same as the one-input model, the more regularization the poorer the performance. The training and validation accuracy show no clear trend when adding more regularization techniques. The last two networks show no improvement during training illustrated in Figure 30. The implementation of batch normalization to the model results in convergence in both train and validation loss (Figure 31, page 24). Employing batch normalization slightly mitigates overfitting, as evidenced by no increase in the validation loss and the reduced gap between training and validation accuracy. It also has the highest accuracy among all networks and one of the lowest validation losses and is therefore the final best-performing model using two inputs.

Table 12: Performances of the five two-input models in experiment 5.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN basic	0.883	0.994	0.933	0.558
CNN WD	0.657	1.219	0.962	0.483
CNN BN	0.639	1.160	0.782	0.569
CNN BN + DA	1.399	1.341	0.360	0.502
CNN BN + DA + WD	1.416	1.308	0.360	0.476

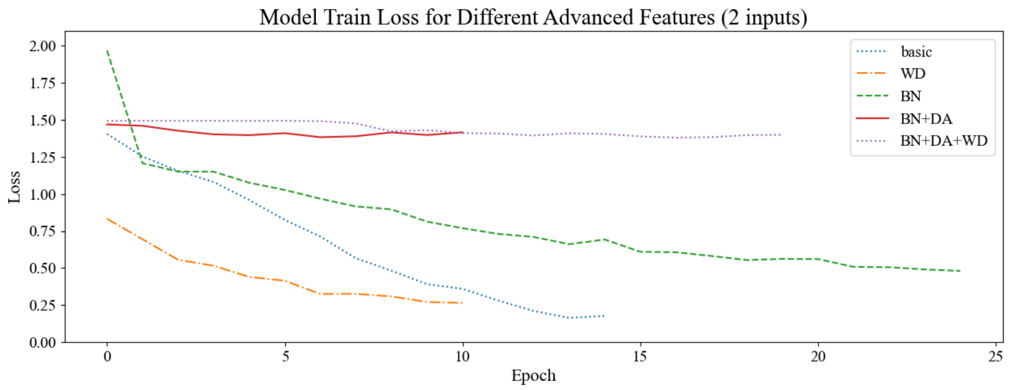


Figure 30: Model train loss over epochs of the five two-input models of experiment 5.

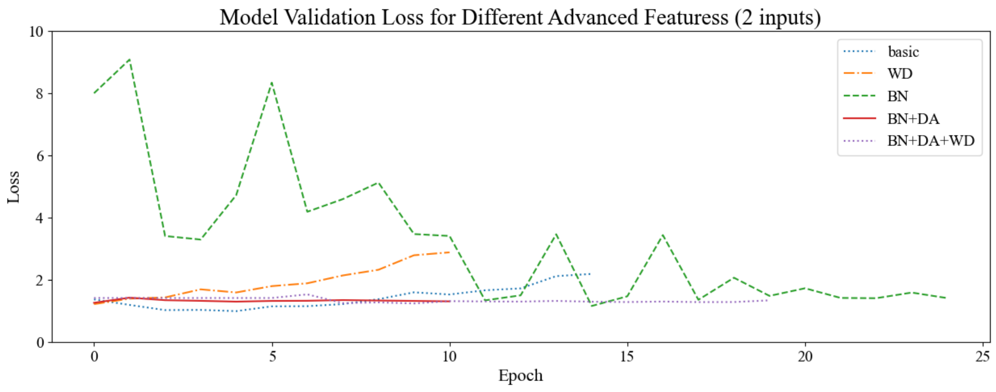


Figure 31: Model validation loss over epochs of the five two-input models of experiment 5.

Batch normalization enhances the performance of single and dual input DCNNs by stabilizing the training process and results in regularization and improved convergence across the different network architectures.

4.2 Best architecture

Out of both single and dual-input models, the model with two input images performs best on the river dataset. Overfitting is diminished and the network has a higher validation accuracy in contrast to the one-input network.

Table 13: Model performances for most optimal single and dual input models.

	Train loss	**Validation loss**	**Train accuracy**	**Validation accuracy**
CNN 1 input	0.388	1.299	0.971	0.546
CNN 2 inputs	0.639	1.160	0.782	0.569

The best architecture for classifying river segments is using two inputs, five convolution and pooling layers, with 24, 48, 96, 192, and 384 feature maps in the convolutional layers, a fully connected layer with 64 dense units, dropouts of 0.2, and batch normalization. The total number of layers results in 26, of which 6 are trainable layers (convolution and fully connected layers). As mentioned in Chapter 3.1, a network having five or more layers and including convolution, pooling, and fully connected layers, the CNN architecture qualifies as a DCNN. Therefore this architecture meets the criteria of a deep convolutional neural network. The final DCNN structure is illustrated in Figures B10 and B11 of Appendix B on pages 44 and 45.

The associated performances of the final model are shown again in Figure 32 on page 25. The validation loss shows significant fluctuations and does not consistently decrease in contrast to the training loss. This divergence is an indicator of overfitting. The validation accuracy also shows fluctuations differently from the training accuracy, indicating the model struggles to generalize well on unseen data. The accuracies continue to improve over epochs however, training was halted by the early stopping mechanism. The implemented early stopping effectively mitigated further overfitting by monitoring the validation loss.

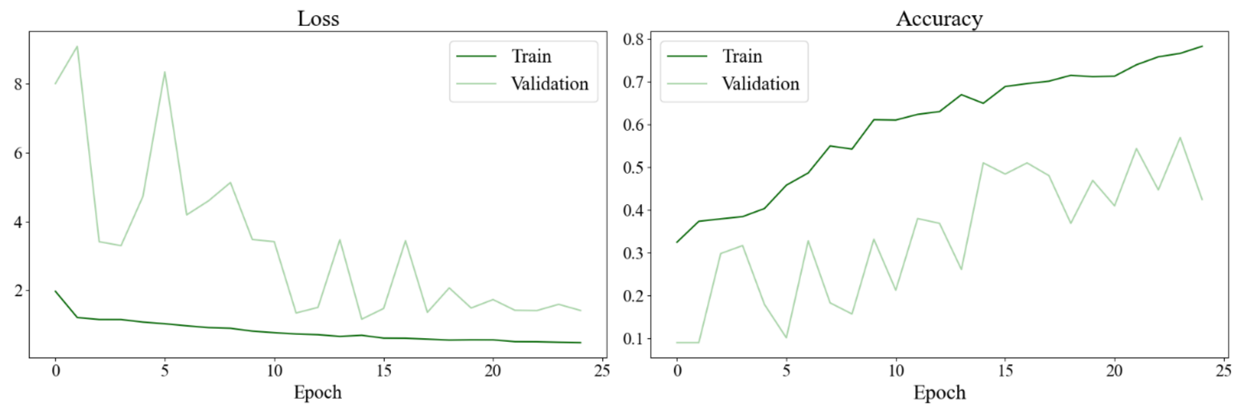


Figure 32: Training and validation metrics over epochs of the final two-input model.

4.3 Testing

The predictions of the final model on the test set are shown in Table 14.

Table 14: Confusion matrix of the predictions on the test set.

		Predicted			
		Anastomosing	Braided	Meandering	Wandering
Actual	Anastomosing	3	1	1	7
	Braided	7	10	0	6
	Meandering	2	1	28	50
	Wandering	0	6	43	104

The corresponding F1-scores of the class anastomosing, braided, meandering, and wandering are 25%, 49%, 37%, and 65%, respectively. The associated weighted F1-score of the DCNN is 53%. As can be seen, the model struggles with classifying anastomosing the most, the majority of anastomosing rivers are classified as wandering. This is likely due to the limited representations of anastomosing rivers in the image dataset. Four river segments that are classified as wandering are illustrated in Figure 33 Reflecting on the initial fraction-labeling of these rivers, the first one in the row is originally labeled as 90% anastomosing and 10% braided with 100% certainty. The second river in the figure has a percentage of 80% anastomosing and 20% meandering with 60% certainty, the third has the same classification combination but with a fraction of 80% and 20% and a lower certainty of 20%. The last river is identified as 100% anastomosing with 100% certainty. Out of all the tested anastomosing rivers, two rivers contain a fraction of a wandering river but one of these is correctly classified as anastomosing, and the other is classified as meandering. The segments labeled as wandering are likely due to it being the majority class, rather than the wandering characteristics in the images.

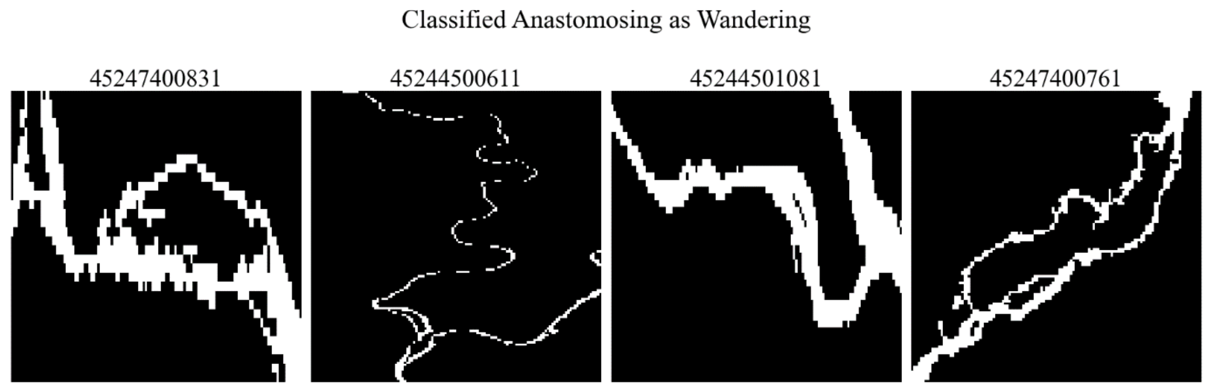


Figure 33: Four examples of anastomosing rivers classified as wandering.

The network confuses braided rivers often with anastomosing, four examples are shown in Figure 34 on page 27. The original labels for the first one and the last two are combinations of anastomosing and braided. 45244501051 and 45246100061 both consist of 20% anastomosing and 80% braided, with 90% and 100% certainty respectively. The last one has 10% anastomosing traits, 90% braided, and is 100% certain. The second one in the row is fully braided. Out of the 23 tested braided rivers nine exhibit fractions of anastomosing and six contain some wandering fractions. Among these six rivers, three are predicted to be wandering. The network's difficulties with distinguishing between braided and anastomosing rivers are not surprising, as humans also find it challenging to differentiate between the two.

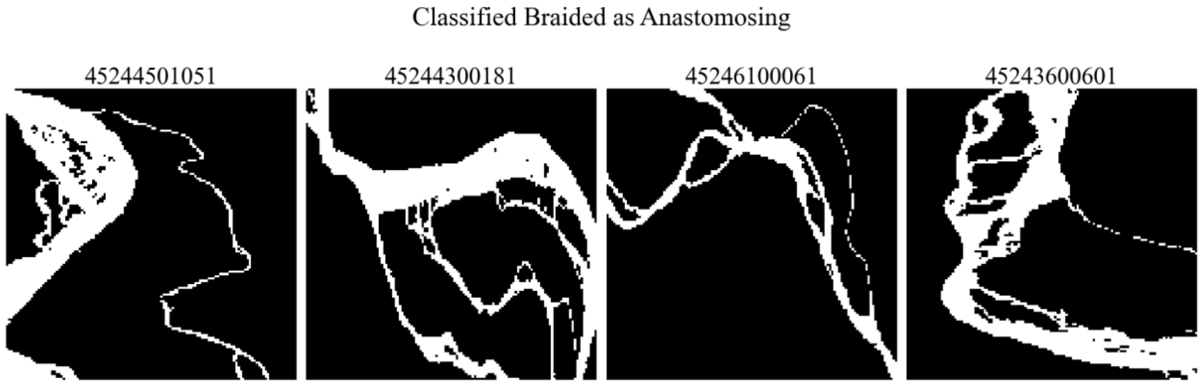


Figure 34: Four examples of braided rivers classified as anastomosing.

The model significantly misclassifies meandering rivers as wandering as can be seen in Table 14 on page 26. 41 of 81 meandering rivers in the test set include wandering features, and out of these are twenty predicted as wandering. Images of four incorrect classifications of meandering are in Figure 35. The first and third pictures are fully meandering with 100% certainties. The second river is 70% meandering and 30% wandering with 70% certainty, and the last image is 20% braided and 80% wandering with 70% certainty.

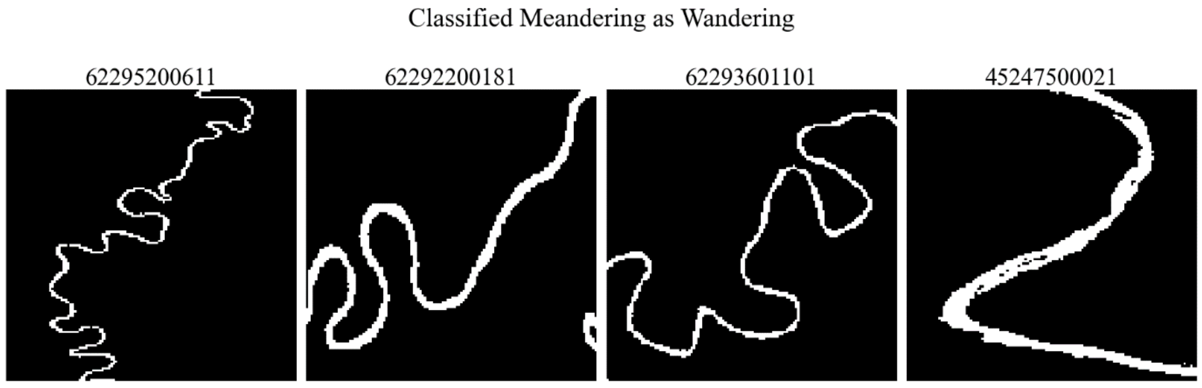


Figure 35: Four examples of meandering rivers classified as wandering.

The DCNN can predict the river morphology wandering with the highest measured F1-score across the four classes considered. When the model misclassifies a wandering river most of the time it is a meandering river. Out of 153 wandering river segments, 34 exhibit meandering characteristics, with twenty classified as meandering. The first, third, and fourth images of Figure 36 are 100% wandering. The first and third have 100% certainty while the certainty for the fourth is not specified. Lastly, the second river segment is 40% meandering and 60% wandering with 60% certainty. The network also struggles to distinguish between meandering and wandering rivers, which is to be expected due to similar appearances.

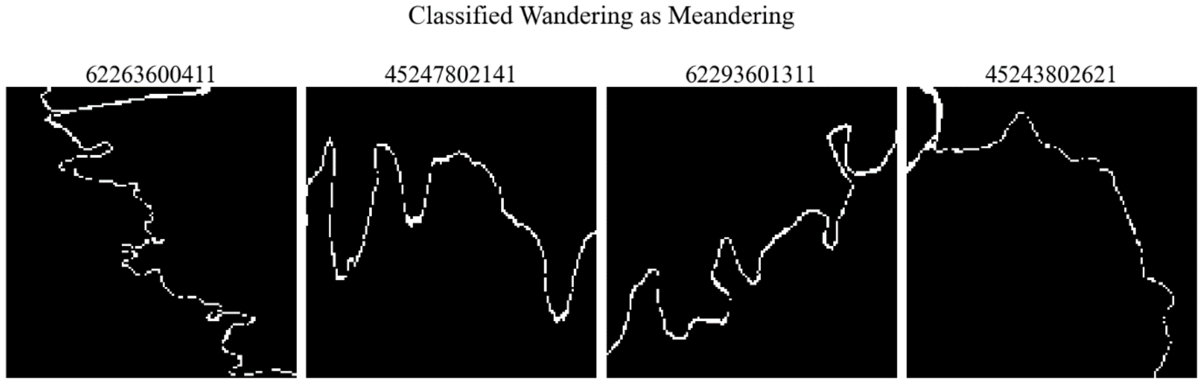


Figure 36: Four examples of wandering rivers classified as meandering.

Five correct classified river segments are presented in Figure 37.

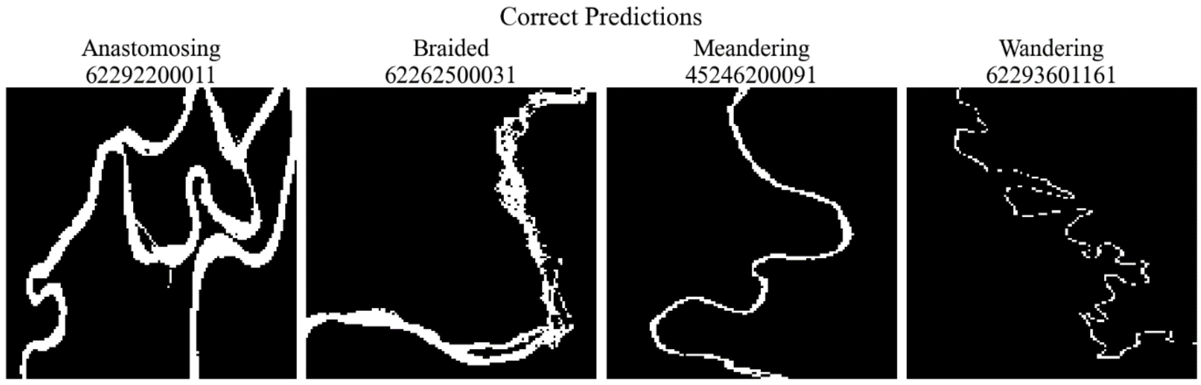


Figure 37: Four examples of correctly classified river segments.

According to the Cohen's d values, the overall performance of the DCNN demonstrates a large improvement compared to the baseline. Classes braided and meandering exhibit the highest effect sizes, indicating that the DCNN model significantly outperforms the baseline in these specific classes. Conversely, for the majority class, where the baseline performs better, the effect size favoring the baseline is small, suggesting the magnitude of this advantage baseline is small compared to the gains of the other classes. This is logical because the baseline model's predictions were primarily based on the majority class, resulting in a higher F1-score for this class by default. Despite wandering being the majority class, the model still struggles with accurately classifying it. The Cohen's d analysis shows the overall effectiveness of the network relative to the baseline.

Table 15: Cohen's d value overall and across river types.

Overall	**Anastomosing**	**Braided**	**Meandering**	**Wandering**
0.9	0.9	1.7	1.3	-0.3

5. Discussion

This chapter reflects on the research process and provides suggestions for further improvement and next steps for future research.

5.1 Dataset

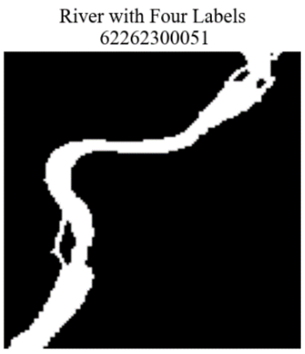


Figure 38: Example of a river with four labels.

Due to the novelty of this research, the dataset required to be manually labeled beforehand. This was challenging because many segments often contained multiple river morphologies. While the images in this thesis mostly showcase clear river types as examples, the data used by the network often did not correspond to one single type. The dataset contains 770 (out of 1,792) river images with two or more labels and even five with four labels. To illustrate, Figure 38 is an example segment that contains all four river types: 20% anastomosing, 10% braided, 35% meandering, and 30% wandering with 30% certainty. Evaluating the images on two separate moments could alter the opinion on what fractions the river type actually has, highlighting the subjective nature of the manual classification. The difficulty in labeling is also evident from the images that were labeled by multiple persons. Out of the 1,033 overlapping labeled images, 673 images did not have the same river type combinations. According to D. Beelen (personal communication, 2024), if four geologists would look at the indistinct images, their answers would probably differ. If the task of classifying river types without additional fluviomorphological parameters is hard for humans, the task is probably also challenging for the DCNN.

The dataset the DCNN used was also limited, in particular, the representation of anastomosing and braided river morphologies was insufficient. To address this class imbalance, Random Oversampling was applied to create equal classes for training. This technique duplicates images of the minority class and does not add new information to the dataset, instead, it reinforces existing patterns. Consequently, the network may learn the patterns too well, which could lead to high performance on training images but poor generalization on unseen data. Despite the use of several regularization techniques to prevent overfitting, the models with one input stay highly overfitted and two inputs slightly overfitted. The small dataset could also be a reason for overfitting. Expanding the dataset, especially for anastomosing and braided rivers could improve the results of the models.

The SWORD database determines reaches based on several factors, such as hydrological conditions. Due to the design of the SWORD database, an anastomosing river might be divided into two separate rivers. In this study, each reach was captured as an individual image for labeling and model input. These images may not indicate anastomosing patterns comprehensively because they represent only parts of the whole river system. Consequently, there might be fewer anastomosing rivers in the dataset. However, the buffer used for clipping the rivers to a single image provided additional information for labeling, which could ensure that the river is correctly identified as anastomosing. As a result, some rivers could still be accurately labeled despite the segmentation into separate reaches. An example image of two separate reaches that belong to one anastomosing river is shown in Figure 39 on the next page. Both reaches (Figures 40 and 41 on page 29) are labeled as anastomosing due to the utilized buffer. However, this does not guarantee that the buffer method consistently provided accurate images for all reaches belonging to anastomosing rivers.

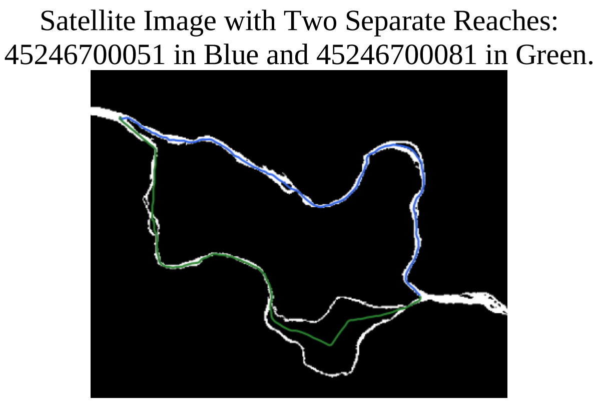

Satellite Image with Two Separate Reaches:
45246700051 in Blue and 45246700081 in Green.

Figure 39: Satellite image of rivers 45246700051 and 45246700081.

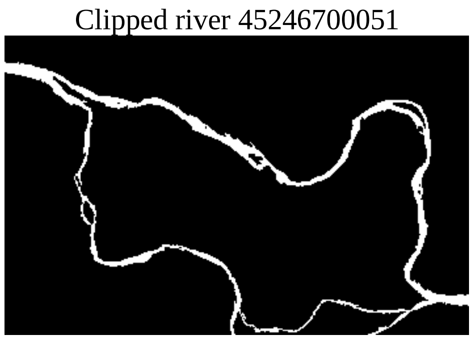


Figure 40: Clipped image of river 45246700051.

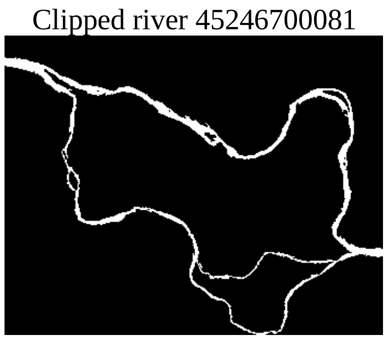


Figure 41: Clipped image of river 45246700081

Approximately half of the labeled river segments contained a certainty measure. To improve model performance, it might benefit from only using the data with certain labels. This however, compromises 254 instances and is not enough to train a DCNN. This further emphasizes the need for an expanded dataset.

5.2 Workflow discussion

Prior to training the DCNNs, during preprocessing, modification was made to the labels of some rivers. When a river segment is over 30% anastomosing or braided and the remainder is meandering or wandering, the river label is set to anastomosing or braided. This could cause inaccurate predictions that are technically also correct. The network might incorrectly classify that type of river as meandering, even though this prediction is correct due to the originally highest fraction. The test dataset did not include examples of anastomosing being a minor fraction, but being the dominant feature after manipulation. Only two rivers were labeled as braided, despite wandering being the predominant characteristic. One of these was correctly classified as braided while the other was incorrectly predicted as wandering. This one was wrongly classified due to manipulation but is technically not wrong. Although the modification ensured compliance with the labeling criteria, they also introduced the possibility of a classification that does not reflect the true nature of the river images.

Another preprocessing step was to only use images with a high resolution. The images smaller than 30 by 30 pixels were deleted from the dataset. The rationale behind this decision was to ensure the DCNN could learn meaningful patterns and extract enough visual information for classification and bias that could arise from inadequate visual information is mitigated by this. In addition, the small river segments could not be labeled by human classifiers.

Despite various experimentations that have been utilized in this study, more experiments could be employed. Due to computational problems during this research, cross-validation to find whether data augmentation was the most optimal for the network could not be utilized and a light data augmentation was applied. Data augmentation generally enhances the performance of the model but in this research,

it did not lead to improved results. Finding the best augmentation, potentially adopting a more aggressive approach (higher parameter values), could lead to better performances.
This study researched data augmentation only in combination with other regularization techniques since the models were extremely overfitting, however, it does not include the optimal model out of the experiments with additional data augmentation during training, which can be tested in addition.

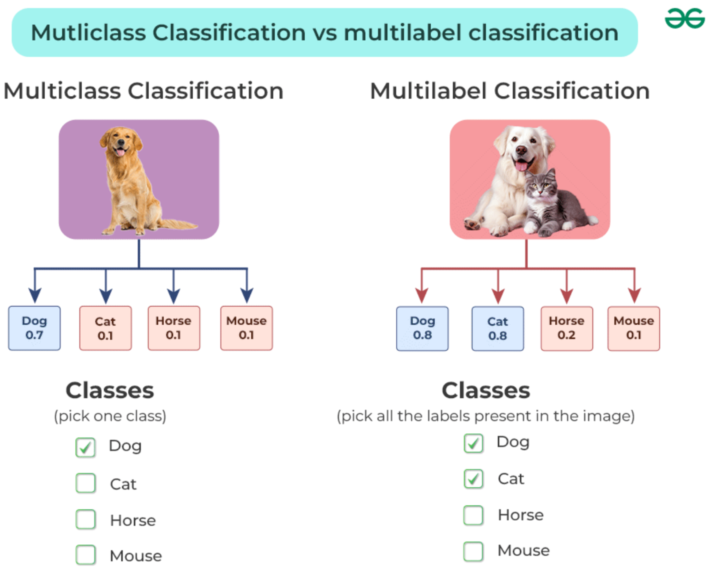
The experiment analyzing the performance of the networks with additional advanced features including weight decay could also involve the fine-tuning of various types of weights. Adjusting weight decay involves experimentation using cross-validation to find the optimal value. A very high weight decay can cause the model to become overly simplistic, leading to information loss, while a very low weight decay can result in overfitting. The weight decay in this study (0.0001) was chosen at a moderate level in comparison to other studies training DCNNs, where weight decay typically varies between 0.00001 and 0.0005 (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Chollet, 2017). Batch normalization could also perform different when placed alternatively. They were currently placed between each convolutional and pooling layer. However, alternative placements or reducing batch normalization layers can be analyzed.

Some employed models in the experiments showed a constant training loss. It is unusual but possible for a model with a specific configuration. However, it most likely suggests issues during training which could be caused by vanishing (too small) or exploding (too large) gradients, which in turn cause the training process to slow down or halt. Furthermore, the considered hyperparameters may interact poorly with the learning rate, resulting in poor convergence. More experimenting with learning rates can mitigate this.

5.3 Future

The limited dataset is a main reason for the modest performance of the network in this study. Labeling more river segments and feeding that to the network would likely improve its performance. As mentioned in Chapter 5.1, and shown in the results, there is a significant demand for anastomosing and braided labeled rivers. When the dataset is larger, the highest label can be set to 1 without manipulation of the minority classes. Starting with doubling the data is advised to gauge improvements in model performance.

Due to the high occurrence of images belonging to multiple categories simultaneously, a multi-label classification neural network, instead of multi-class classification, can be employed to test its performance. These networks can classify one image with more labels, as shown in Figure 42, and could be a beneficial approach. The model can handle more complex and realistic scenarios compared to a single-label classification model. A big difference with the DCNN in this study and a multi-label model is a different activation function and loss function. Instead of a softmax activation function in the output layer and categorical cross-entropy loss, a sigmoid activation function, and binary cross-entropy are assigned to the network (Pan et al., 2019; Wang et al., 2023). Binary relevance is required since each label is treated independently, present or


Mutliclass Classification vs multilabel classification
Multiclass Classification
Multilabel Classification
Dog 0.7
Cat 0.1
Horse 0.1
Mouse 0.1
Dog 0.8
Cat 0.8
Horse 0.2
Mouse 0.1
Classes
(pick one class)
Dog
Cat
Horse
Mouse
Classes
(pick all the labels present in the image)
Dog
Cat
Horse
Mouse
Figure 42: Difference between multi-class and multi-label classification (GeeksforGeeks, 2024)

not present, and the network therefore has separate outputs for each river morphology. A binary loss function needs a sigmoid as activation since each class has independent probabilities that do not need to sum up to one (also shown in Figure 42).
Multi-label image classification models have already been utilized in research involving remote sensing images for land cover classifications. Cheng et al. (2024) proposed novel methods that include adopting the ResNet architecture for deep feature extraction. Their Multi-Band Multi-Label network achieves a top accuracy of 83.52% and a corresponding F1-score of 77.97% and suppresses other deep learning models and conventional methods, demonstrating exceptional performance in multispectral multi-label classification of remote sensing imagery. In addition, You et al. (2023), present a deep learning model based on a highly connected CNN enhanced with a Convolutional Block Attention Module. This model is based on DenseNet which reduces parameters compared to ResNet. The experimental results show that the model achieves a 1.2% higher accuracy than an existing network, DenseNet121, and a micro F1-score of 92%. It effectively improves the classification accuracies of remote sensing images.

6. Conclusion

This thesis researched the capability of a (Deep) Convolutional Neural Network to accurately classify the four different river morphology types, as well as identifying the optimal model architecture for this specific classification task.

Before employing DCNNs, each river segment is manually labeled by four persons. The labeled dataset has unequal classes and thus Random Oversampling was applied to the training dataset to create balanced river classes. Architectures have been employed with satellite images as sole input and networks with additional images of the corresponding centerlines of the rivers. Various experiments have been conducted to determine the most optimal architecture.

The optimal architecture is a two-input DCNN having five convolutional and pooling layers, with 24, 48, 96, 192, and 384 feature maps in the convolutional layers. It includes a fully connected layer with 64 dense units, dropouts of 0.2 after each pooling layer and between the fully connected- and the output layer, and batch normalization after the convolutional layers. The weighted F1-score is 53% and the F1-score for river morphologies anastomosing, braided, meandering, and wandering are 25%, 49%, 37%, and 65%, respectively. The model is still slightly overfitted despite the use of regularization techniques. The network struggles with distinguishing braided and anastomosing rivers alongside with wandering and meandering. The DCNN significantly outperforms the baseline model that predicts the majority class. A Cohen's d value of 0.9 indicates a substantial improvement of the DCNN over the baseline. The network can find meaningful patterns and shows effectiveness in distinguishing between classes. These results are therefore a meaningful achievement given the novelty of the task.

Expanding the river dataset with additional labels, especially for anastomosing and braided rivers would enable the models to learn more features and potentially enhance their performances and mitigate the overfitting issue. It is recommended to begin by doubling the data to evaluate enhancements in model performance.

Further improvement and optimization can be utilized to achieve higher performances in future work. Applying a multi-label network could be beneficial due to the frequent occurrence of multiple morphologies within each image.

This study marked the first attempt and demonstrated the feasibility and potential of classifying river morphologies using DCNNs.

References

Adegun, A. A., Viriri, S., & Tapamo, J. (2023). Review of deep learning methods for remote sensing satellite images classification: experimental survey and comparative analysis. *Journal of Big Data*, *10*(1). https://doi.org/10.1186/s40537-023-00772-x

Alabyan, A. M., & Chalov, R. S. (1998). Types of river channel patterns and their natural controls. *Earth Surface Processes and Landforms*, *23*(5), 467–474. https://doi.org/10.1002/(sici)1096-9837(199805)23:5

Altenau, E. H., Pavelsky, T. M., Durand, M., Yang, X., De Moraes Frasson, R. P., & Bendezu, L. (2021). The Surface Water and Ocean Topography (SWOT) Mission River Database (SWORD): a global river network for satellite data products. *Water Resources Research*, *57*(7). https://doi.org/10.1029/2021wr030054

Baguley, T. (2009). Standardized or simple effect size: What should be reported? *British Journal of Psychology*, *100*(3), 603–617. https://doi.org/10.1348/000712608x377117

Basha, S. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378, 112–119. https://doi.org/10.1016/j.neucom.2019.10.008

Berngardt, O., I. (2024). Minimum number of neurons in fully connected layers of a given neural network (the first approximation). *arXiv (Cornell University).* https://doi.org/10.48550/arxiv.2405.14147

Biancamaria, S., Lettenmaier, D. P., & Pavelsky, T. M. (2016). The SWOT mission and its capabilities for land hydrology. In *Space sciences series of ISSI* (pp. 117–147). https://doi.org/10.1007/978-3-319-32449-4_6

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Neural Information Processing Systems*, *31*, 7694–7705. https://papers.nips.cc/paper/7996-understanding-batch-normalization.pdf

Cai, S., Gao, J., Zhang, M., Wang, W., Chen, G., & Ooi, B. C. (2019). Effective and efficient dropout for deep convolutional neural networks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1904.03392

Charlton, R. (2008). *Fundamentals of Fluvial Geomorphology*. Routledge. https://text2fa.ir/wp-content/uploads/Text2fa.ir-FUNDAMENTALS-OF-FLUVIAL-1.pdf

Chen, M. (2021). Convolutional Neural Network with Pruning Method for Handwritten Digit Recognition. *arXiv (Cornell University*). https://doi.org/10.48550/arxiv.2101.05996

Cheng, X., Li, B., Deng, Y., Tang, J., Shi, Y., & Zhao, J. (2024). MMDL-NET: Multi-Band Multi-Label Remote Sensing Image Classification Model. *Applied Sciences*, 14(6), 2226. https://doi.org/10.3390/app14062226

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* https://doi.org/10.1109/cvpr.2017.195

Cohen, J. (2013). Statistical Power Analysis for the Behavioral Sciences. In *Routledge eBooks*. https://doi.org/10.4324/9780203771587

Dalianis, H. (2018). Clinical text mining. In *Springer eBooks*. https://doi.org/10.1007/978-3-319-78503-5

De Moraes Frasson, R. P., Pavelsky, T. M., Fonstad, M. A., Durand, M., Allen, G. H., Schumann, G., Lion, C., Beighley, R. E., & Yang, X. (2019). Global relationships between river width, slope, catchment area, meander wavelength, sinuosity, and discharge. *Geophysical Research Letters*, *46*(6), 3252–3262. https://doi.org/10.1029/2019gl082027

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf

Google Maps. (n.d.-a) [Example of an Anastomosing river, the Ob River in Russia] https://www.google.com/maps/@61.2453859,71.1039675,43378m/data=!3m1!1e3?hl=nl&entry=ttu

Google Maps. (n.d.-b) [Example of a Braided river, the Brahmaputra River in Asia] https://www.google.com/maps/@26.1794279,91.2861,26810m/data=!3m1!1e3?hl=nl&entry=ttu

Google Maps. (n.d.-c) [Example of a Meandering river, the Curuçá River in Brazil] https://www.google.com/maps/@-4.99097,-71.804664,24490m/data=!3m1!1e3?hl=nl&entry=ttu

Google Maps. (n.d.-d) [Example of a Wandering river, the Barwon River in Austria] https://www.google.nl/maps/place/Barwon+River/@-38.1306186,144.2039536,7094m/data=!3m1!1e3!4m6!3m5!1s0x6ad43f7d722391ef:0xfdfdf3870f3e0f37!8m2!3d-38.260563!4d144.4878866!16zL20vMDNrd3d3!5m1!1e4?hl=nl&entry=ttu

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

Harvey, B. (2024). *Principles of Deep Learning in Artificial Networks* [Slide show].

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv (Cornell University).* https://doi.org/10.48550/arxiv.1502.03167

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Deep Learning and Data Sampling with Imbalanced Big Data. *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. https://doi.org/10.1109/iri.2019.00038

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, 25, 1097–1105. http://books.nips.cc/papers/files/nips25/NIPS2012_0534.pdf

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. In *Springer eBooks.* https://doi.org/10.1007/978-1-4614-6849-3

Lancaster, S. T., & Bras, R. L. (2002). A simple model of river meandering and its comparison to natural channels. *Hydrological Processes*, *16*(1), 1–26. https://doi.org/10.1002/hyp.273

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Leopold, L. B., & Wolman, M. G. (1957). River channel patterns: Braided, meandering, and straight. *U.S. Geological Survey Professional Papers/U.S. Geological Survey Professional Paper*. https://doi.org/10.3133/pp282b

Li, G., Jian, X., Wen, Z., & AlSultan, J. (2022). Algorithm of overfitting avoidance in CNN based on maximum pooled and weight decay. *Applied Mathematics and Nonlinear Sciences*, *7*(2), 965–974. https://doi.org/10.2478/amns.2022.1.00011

Li, Z., Yan, C., & Boota, M. W. (2022). Review and outlook of river morphology expression. *Journal of Water and Climate Change*, *13*(4), 1725–1747. https://doi.org/10.2166/wcc.2022.449

Lindsay, G. W. (2021). Convolutional neural networks as a model of the visual system: past, present, and future. *Journal of Cognitive Neuroscience*, *33*(10), 2017–2031. https://doi.org/10.1162/jocn_a_01544

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences/Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150203. https://doi.org/10.1098/rsta.2015.0203

Manning, A. J. (Ed.) (2022). River Deltas Research - Recent advances. In *IntechOpen eBooks*. https://doi.org/10.5772/intechopen.78857

Morisawa, M. (1989). Rivers and valleys of Pennsylvania, revisited. *Geomorphology*, *2*(1–3), 1–22. https://doi.org/10.1016/0169-555x(89)90003-2

Nath, A., & Ghosh, S. (2022). Assessment of river morphology based on changes in land use and land cover and the spatial and temporal variation of meandering parameters of the barak river. *Water Practice and Technology*, *17*(11), 2351–2370. https://doi.org/10.2166/wpt.2022.114

Nyberg, B., Henstra, G. A., Gawthorpe, R. L., Ravnås, R., & Ahokas, J. (2023). Global scale analysis on the extent of river channel belts. *Nature Communications*, *14*(1). https://doi.org/10.1038/s41467-023-37852-8

Oga, T., Harakawa, R., Minewaki, S., Umeki, Y., Matsuda, Y., & Iwahashi, M. (2020). River state classification combining patch-based processing and CNN. *PloS One*, *15*(12), e0243073. https://doi.org/10.1371/journal.pone.0243073

Pan, D., Lu, Y., & Kang, P. (2019). A deep learning model for multi-label classification using capsule networks. *In Lecture notes in computer science* (pp. 144–155). https://doi.org/10.1007/978-3-030-26763-6_14

Podareanu, D., Codreanu, V., Van Leeuwen, C., Aigner, S., & Weinberg, V. (2019). Best Practice Guide - Deep Learning. In *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.5281/zenodo.4700781

Pratiwi, N. K. C., Fu'adah, Y. N., & Edwar, E. (2021). Early Detection of Deforestation through Satellite Land Geospatial Images based on CNN Architecture. *Jurnal Infotel/Jurnal Infotel*, *13*(2), 54–62. https://doi.org/10.20895/infotel.v13i2.642

Qui˜nonero-Candela, J., Rasmussen, C. E., Sinz, F., Bousquet, O., & Sch¨Olkopf, B. (2006). Evaluating predictive uncertainty challenge. In *PASCAL* (pp. 1–27) [Book-chapter]. Springer-Verlag Berlin Heidelberg. https://quinonero.net/Publications/quinonero06epuc.pdf

Rebuffi, S., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., & Mann, T. A. (2021). Data augmentation can improve robustness. *arXiv (Cornell University),* 34. https://arxiv.org/pdf/2111.05328

Reynolds, & John Emslie (1851). Panoramic plan of the principal rivers and lakes.

Reza, M. S., & Ma, J. (2018). Imbalanced Histopathological Breast Cancer Image Classification with Convolutional Neural Network. *2018 14th IEEE International Conference on Signal Processing (ICSP)*. https://doi.org/10.1109/icsp.2018.8652304

Sasada, T., Liu, Z., Baba, T., Hatano, K., & Kimura, Y. (2020). A resampling method for imbalanced datasets considering noise and overlap. *Procedia Computer Science*, 176, 420–429. https://doi.org/10.1016/j.procs.2020.08.043

Saxena, A. (2022). An introduction to convolutional neural networks. *International Journal for Research in Applied Science and Engineering Technology*, *10*(12), 943–947. https://doi.org/10.22214/ijraset.2022.47789

Schumm, S. A. (1985). PATTERNS OF ALLUVIAL RIVERS. *Annual Review of Earth and Planetary Sciences*, *13*(1), 5–27. https://doi.org/10.1146/annurev.ea.13.050185.000253

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for Large-Scale image recognition. *Computer Vision and Pattern Recognition*. http://export.arxiv.org/pdf/1409.1556

Tahir, M., Naeem, A., Malik, H., Tanveer, J., Naqvi, R. A., & Lee, S. (2023). DSCC_NET: Multi-Classification Deep Learning Models for diagnosing of skin Cancer using Dermoscopic Images. *Cancers*, 15(7), 2179. https://doi.org/10.3390/cancers15072179

Thambawita, V., Strümke, I., Hicks, S. A., Halvorsen, P., Parasa, S., & Riegler, M. A. (2021). Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images. *Diagnostics*, *11*(12), 2183. https://doi.org/10.3390/diagnostics11122183

Wang, F., Mizrachi, S., Beladev, M., Nadav, G., Amsalem, G., Assaraf, K. L., & Boker, H. H. (2023). MuMIC – Multimodal Embedding for Multi-Label Image Classification with Tempered Sigmoid. *Proceedings of the . . . AAAI Conference on Artificial Intelligence*, 37(13), 15603–15611. https://doi.org/10.1609/aaai.v37i13.26850

Yamashita, R., Nishio, M., Gian, R. K., DO, & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights Into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Yao, X., Yang, H., Wu, Y., Wu, P., Wang, B., Zhou, X., & Wang, S. (2019). Land use classification of the deep convolutional neural network method reducing the loss of spatial features. *Sensors*, *19*(12), 2792. https://doi.org/10.3390/s19122792

You, H., Gu, J., & Jing, W. (2023). Multi-Label remote sensing image land cover classification based on a Multi-Dimensional Attention Mechanism. *Remote Sensing*, 15(20), 4979. https://doi.org/10.3390/rs15204979

Appendix A

This appendix contains example images, a more detailed explanation of the applied parameters, and proof of the baseline F1-scores.

Unusable images by SMOTE Tomek

The figure below shows unusable images generated by SMOTE Tomek. The images contain multiple rivers overlaying each other in grey which is inapplicable for training the (D)CNN.

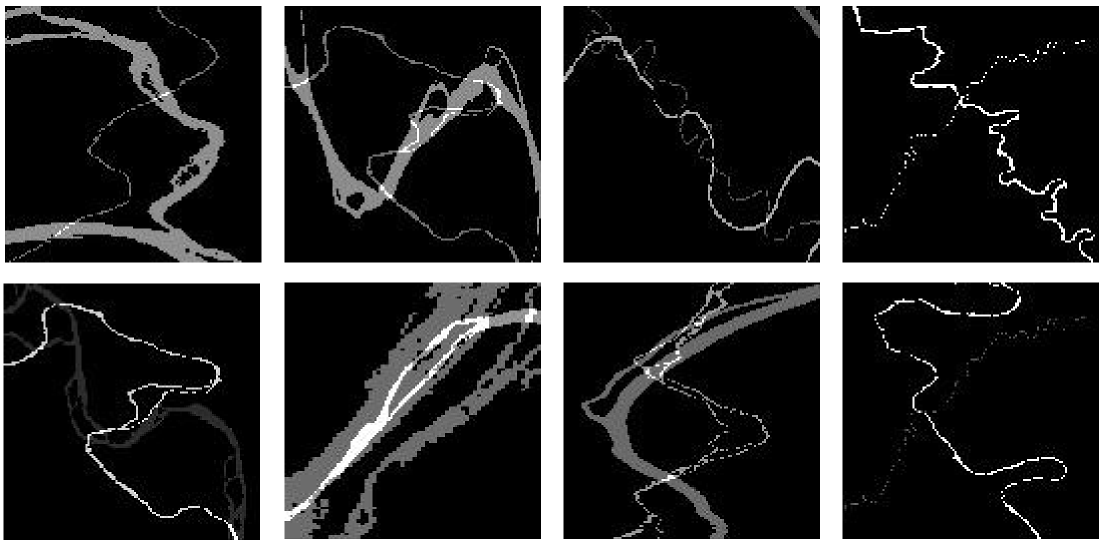


Figure A1: Eight example images of badly generated images by SMOTE.

Data augmentation

The data augmentation that is used in this thesis is shown in Table A1. The rotation range is the range within which random rotations can be employed to the image (e.g. 10 means rotate by any angle between -10 or +10 degrees). Width and height shift range control the range that the image can be horizontally or vertically shifted (e.g. 0.3 means shifted by a maximum of 30% left or right/up- or downwards). Shear range specifies to what extent a images is sheared, meaning the image looks like it is tilted. (e.g. 0.2 indicates the image is sheared up by 20% along their axis. Zoom range defines the range of zooming in or out (e.g. 0.1. can be in or out zoomed up to 10%) Lastly, the horizontal flip is when the image is mirrored along the horizontal axis.

Table A1: Employed data augmentation parameters.

Rotation range	**Width shift range**	**Height shift range**	**Shear range**	**Zoom range**	**Horizontal flip**
10	0.1	0.1	0.1	0.1	True

Proof F1-score by predicting majority class

F1-score per class

Given 12, 23, 81, 153 samples for anastomosing, braided, meandering, and wandering in the test dataset.

Anastomosing: $\text{Precision} = \frac{0}{0+0} = 0, \text{Recal} = \frac{0}{0+12} = 0, \text{F1} - \text{score} = 0$

Braided: $\text{Precision} = \frac{0}{0+0}, \text{Recal} = \frac{0}{0+23} = 0, \text{F1} - \text{score} = 0$

Meandering: $\text{Precision} = \frac{0}{0+0}, \text{Recal} = \frac{0}{0+81} = 0, \text{F1} - \text{score} = 0$

Wandering: Precision $= \frac{153}{153+116} = \frac{153}{269}$, Recal $= \frac{153}{153+0} = 1, \mathrm{F1-score} = \ 2 \times \frac{\frac{153}{269} \times 1}{\frac{153}{269} + 1} \approx 0.73$

Weighted F1-score

Since the F1-score for all classes is equal, the overall F1-score is the average:

$$\text{Weighted F1} - \text{score} = \sum_{i=1}^{4} w_i \ \times F1 - score_i$$
$$= \left(\frac{12}{269} \times 0\right) + \left(\frac{23}{269} \times 0\right) + \left(\frac{81}{269} \times 0\right) + \left(\frac{153}{269} \times \frac{153}{211}\right) \approx 0.41$$

Where

w_i: weight of class i, proportion of the true instances of class i in test dataset.

Appendix B

This appendix shows additional graphs to support the results of the created models.

Convolutional layers

One-input model

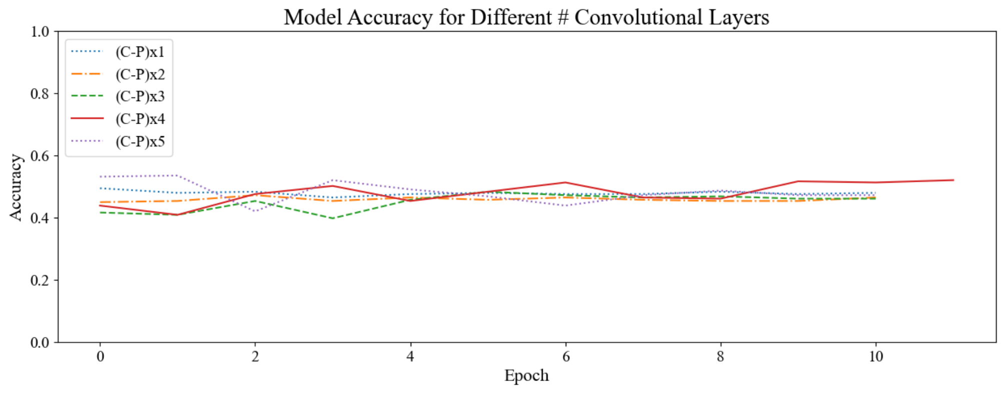


Figure B1: Model accuracy over epochs of the five one-input models of experiment 1.

Two-input model

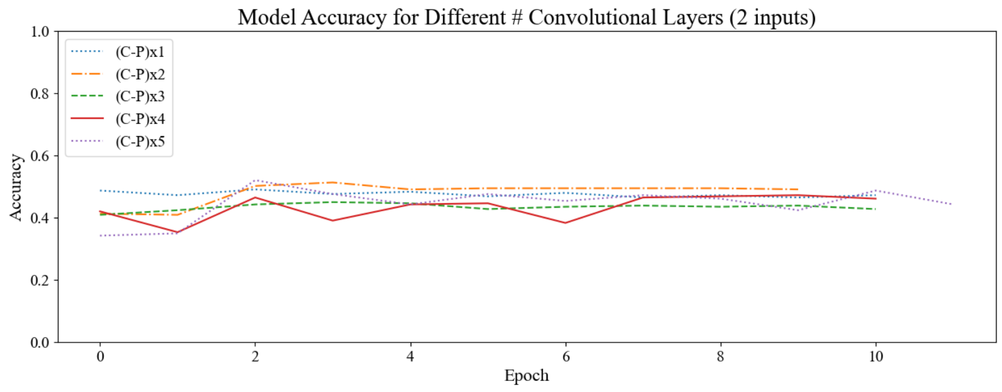


Figure B2: Model accuracy over epochs of the five two-input models of experiment 1.

Feature maps

One-input model

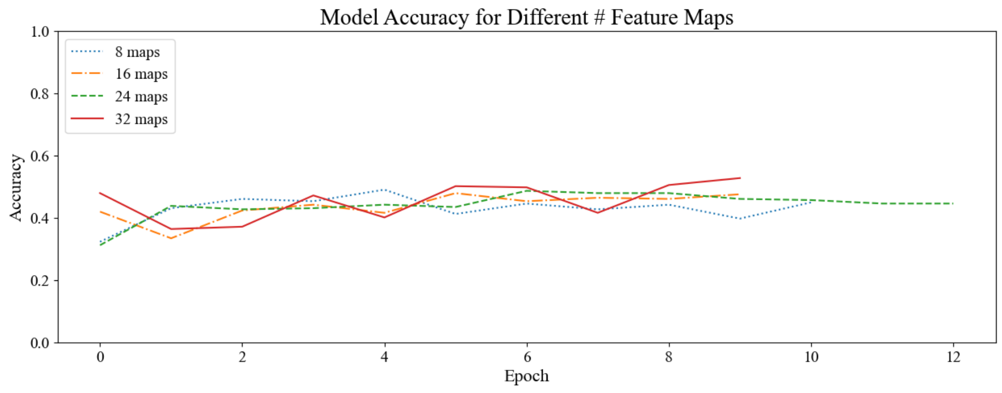


Figure B3: Model accuracy over epochs of the four one-input models of experiment 2.

Two-input model

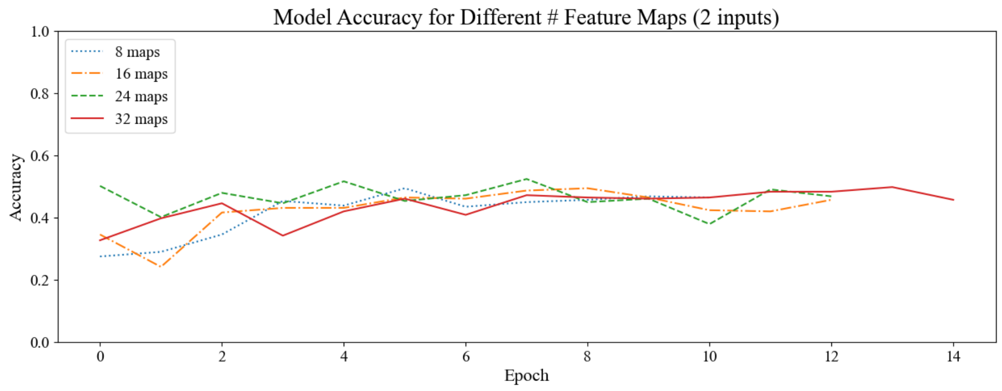


Figure B4: Model accuracy over epochs of the four two-input models of experiment 2.

Dense units

One-input model

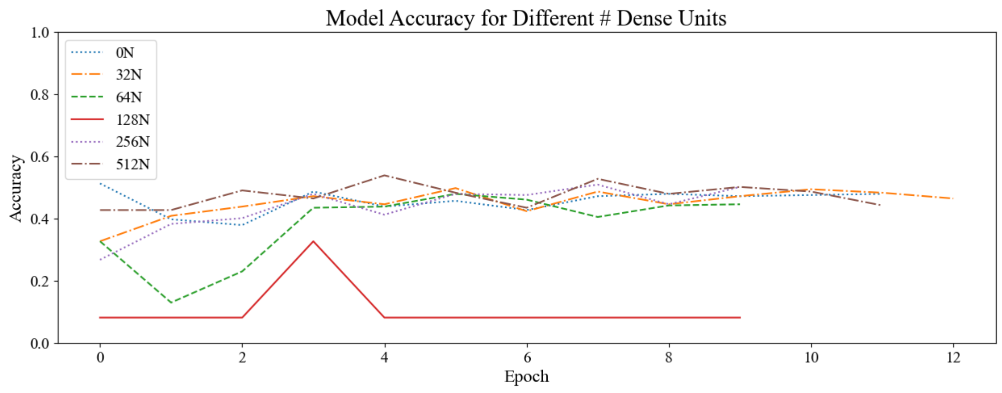


Figure B5: Model accuracy over epochs of the six one-input models of experiment 3.

Two-input model

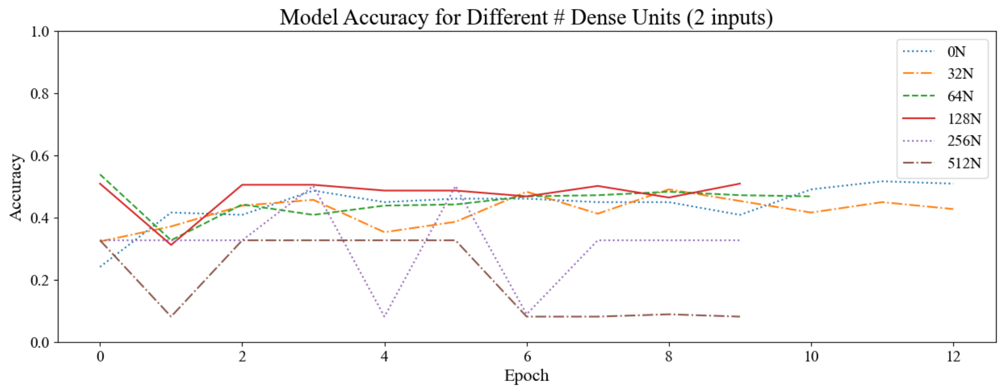


Figure B6: Model accuracy over epochs of the six two-input models of experiment 3.

Dropout rate

One-input model

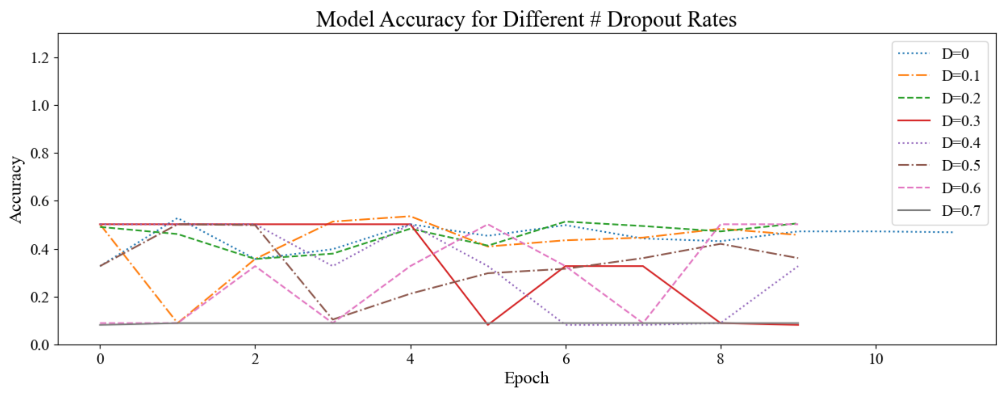


Figure B7: Model accuracy over epochs of the eight one-input models of experiment 4.

Two-input model

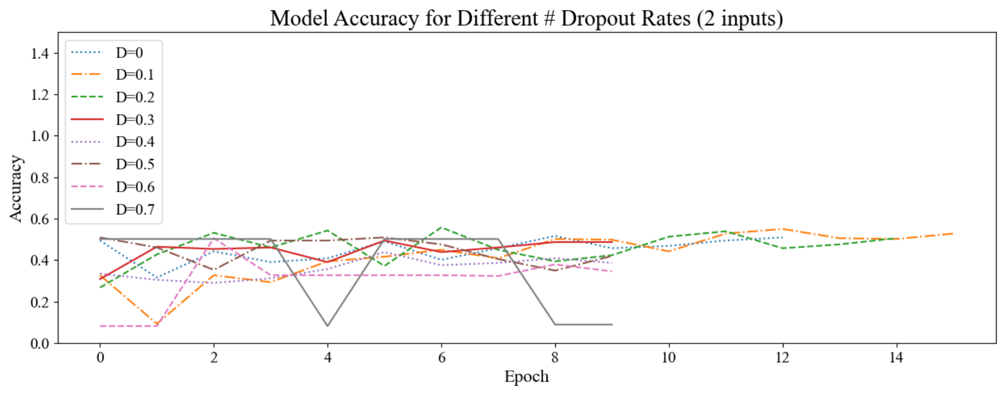


Figure B7: Model accuracy over epochs of the eight two-input models of experiment 4.

Advanced features

One-input model

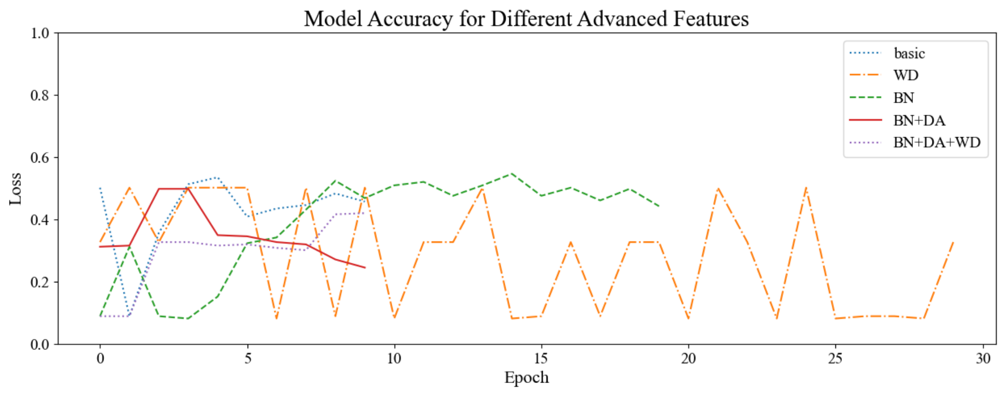


Figure B8: Model accuracy over epochs of the five one-input models of experiment 5.

Two-input model

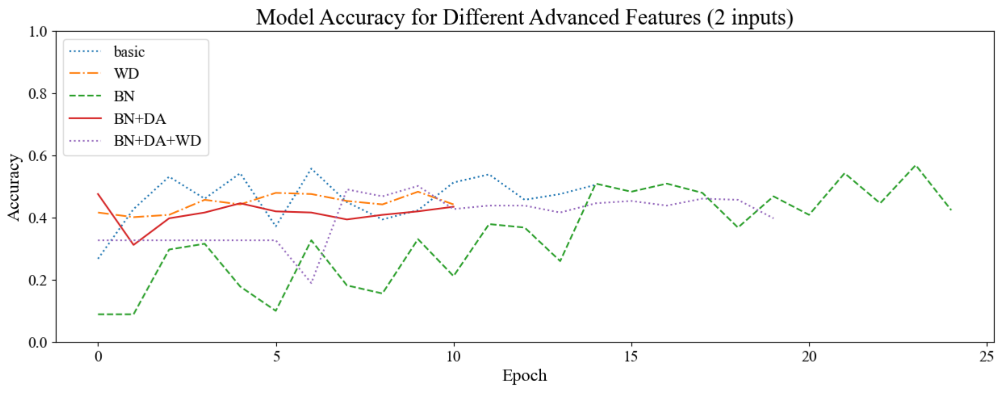


Figure B9: Model accuracy over epochs of the five two-input models of experiment 5.

Best model architecture

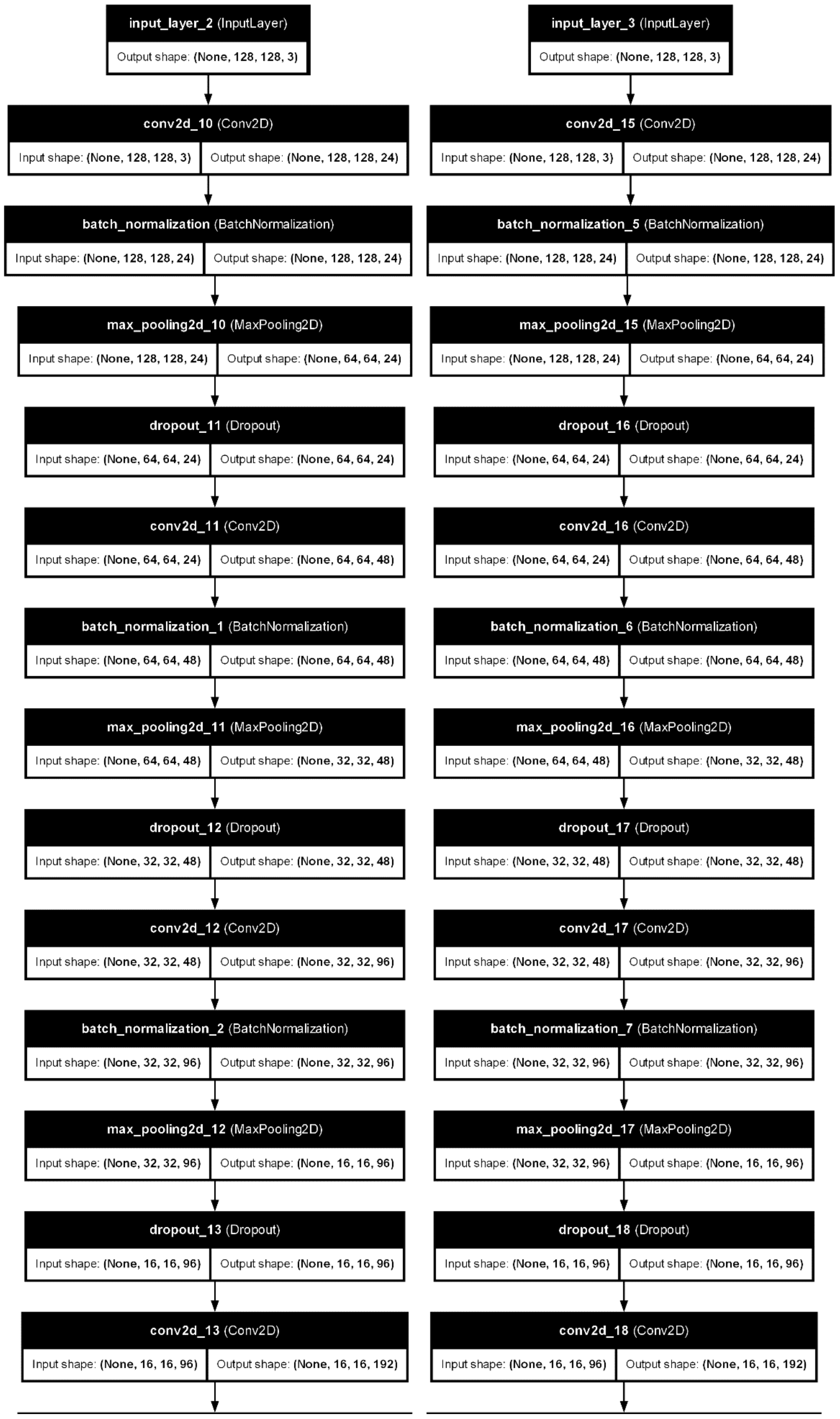
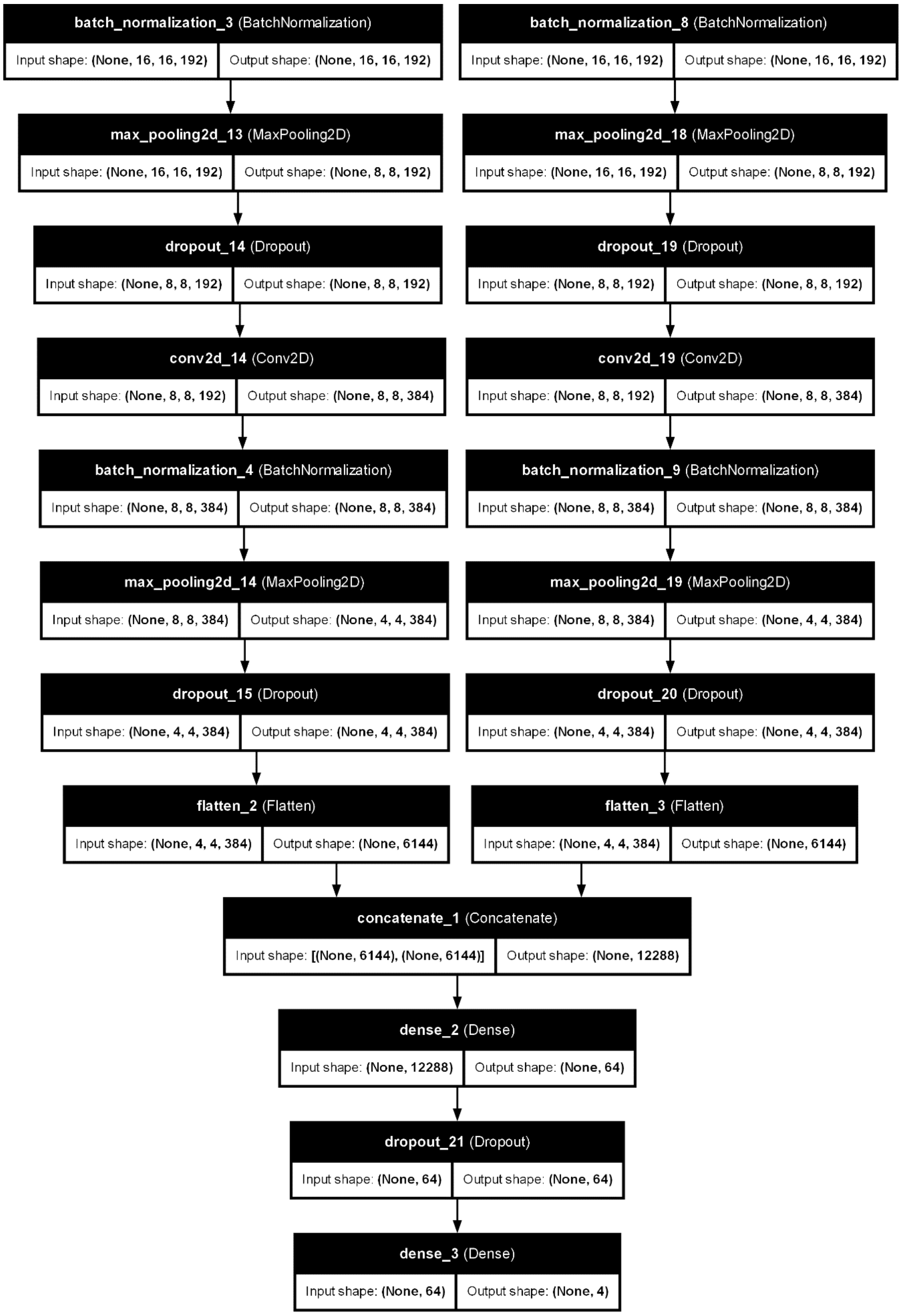

input_layer_2 (InputLayer)
Output shape: (None, 128, 128, 3)
conv2d_10 (Conv2D)
Input shape: (None, 128, 128, 3)
Output shape: (None, 128, 128, 24)
batch_normalization (BatchNormalization)
Input shape: (None, 128, 128, 24)
Output shape: (None, 128, 128, 24)
max_pooling2d_10 (MaxPooling2D)
Input shape: (None, 128, 128, 24)
Output shape: (None, 64, 64, 24)
dropout_11 (Dropout)
Input shape: (None, 64, 64, 24)
Output shape: (None, 64, 64, 24)
conv2d_11 (Conv2D)
Input shape: (None, 64, 64, 24)
Output shape: (None, 64, 64, 48)
batch_normalization_1 (BatchNormalization)
Input shape: (None, 64, 64, 48)
Output shape: (None, 64, 64, 48)
max_pooling2d_11 (MaxPooling2D)
Input shape: (None, 64, 64, 48)
Output shape: (None, 32, 32, 48)
dropout_12 (Dropout)
Input shape: (None, 32, 32, 48)
Output shape: (None, 32, 32, 48)
conv2d_12 (Conv2D)
Input shape: (None, 32, 32, 48)
Output shape: (None, 32, 32, 96)
batch_normalization_2 (BatchNormalization)
Input shape: (None, 32, 32, 96)
Output shape: (None, 32, 32, 96)
max_pooling2d_12 (MaxPooling2D)
Input shape: (None, 32, 32, 96)
Output shape: (None, 16, 16, 96)
dropout_13 (Dropout)
Input shape: (None, 16, 16, 96)
Output shape: (None, 16, 16, 96)
conv2d_13 (Conv2D)
Input shape: (None, 16, 16, 96)
Output shape: (None, 16, 16, 192)
input_layer_3 (InputLayer)
Output shape: (None, 128, 128, 3)
conv2d_15 (Conv2D)
Input shape: (None, 128, 128, 3)
Output shape: (None, 128, 128, 24)
batch_normalization_5 (BatchNormalization)
Input shape: (None, 128, 128, 24)
Output shape: (None, 128, 128, 24)
max_pooling2d_15 (MaxPooling2D)
Input shape: (None, 128, 128, 24)
Output shape: (None, 64, 64, 24)
dropout_16 (Dropout)
Input shape: (None, 64, 64, 24)
Output shape: (None, 64, 64, 24)
conv2d_16 (Conv2D)
Input shape: (None, 64, 64, 24)
Output shape: (None, 64, 64, 48)
batch_normalization_6 (BatchNormalization)
Input shape: (None, 64, 64, 48)
Output shape: (None, 64, 64, 48)
max_pooling2d_16 (MaxPooling2D)
Input shape: (None, 64, 64, 48)
Output shape: (None, 32, 32, 48)
dropout_17 (Dropout)
Input shape: (None, 32, 32, 48)
Output shape: (None, 32, 32, 48)
conv2d_17 (Conv2D)
Input shape: (None, 32, 32, 48)
Output shape: (None, 32, 32, 96)
batch_normalization_7 (BatchNormalization)
Input shape: (None, 32, 32, 96)
Output shape: (None, 32, 32, 96)
max_pooling2d_17 (MaxPooling2D)
Input shape: (None, 32, 32, 96)
Output shape: (None, 16, 16, 96)
dropout_18 (Dropout)
Input shape: (None, 16, 16, 96)
Output shape: (None, 16, 16, 96)
conv2d_18 (Conv2D)
Input shape: (None, 16, 16, 96)
Output shape: (None, 16, 16, 192)

Figure B10: Upper half of the final (D)CNN architecture.



batch_normalization_3 (BatchNormalization)
Input shape: (None, 16, 16, 192)
Output shape: (None, 16, 16, 192)
max_pooling2d_13 (MaxPooling2D)
Input shape: (None, 16, 16, 192)
Output shape: (None, 8, 8, 192)
dropout_14 (Dropout)
Input shape: (None, 8, 8, 192)
Output shape: (None, 8, 8, 192)
conv2d_14 (Conv2D)
Input shape: (None, 8, 8, 192)
Output shape: (None, 8, 8, 384)
batch_normalization_4 (BatchNormalization)
Input shape: (None, 8, 8, 384)
Output shape: (None, 8, 8, 384)
max_pooling2d_14 (MaxPooling2D)
Input shape: (None, 8, 8, 384)
Output shape: (None, 4, 4, 384)
dropout_15 (Dropout)
Input shape: (None, 4, 4, 384)
Output shape: (None, 4, 4, 384)
flatten_2 (Flatten)
Input shape: (None, 4, 4, 384)
Output shape: (None, 6144)
batch_normalization_8 (BatchNormalization)
Input shape: (None, 16, 16, 192)
Output shape: (None, 16, 16, 192)
max_pooling2d_18 (MaxPooling2D)
Input shape: (None, 16, 16, 192)
Output shape: (None, 8, 8, 192)
dropout_19 (Dropout)
Input shape: (None, 8, 8, 192)
Output shape: (None, 8, 8, 192)
conv2d_19 (Conv2D)
Input shape: (None, 8, 8, 192)
Output shape: (None, 8, 8, 384)
batch_normalization_9 (BatchNormalization)
Input shape: (None, 8, 8, 384)
Output shape: (None, 8, 8, 384)
max_pooling2d_19 (MaxPooling2D)
Input shape: (None, 8, 8, 384)
Output shape: (None, 4, 4, 384)
dropout_20 (Dropout)
Input shape: (None, 4, 4, 384)
Output shape: (None, 4, 4, 384)
flatten_3 (Flatten)
Input shape: (None, 4, 4, 384)
Output shape: (None, 6144)
concatenate_1 (Concatenate)
Input shape: [(None, 6144), (None, 6144)]
Output shape: (None, 12288)
dense_2 (Dense)
Input shape: (None, 12288)
Output shape: (None, 64)
dropout_21 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
dense_3 (Dense)
Input shape: (None, 64)
Output shape: (None, 4)

Figure B11: Bottom half of the final (D)CNN architecture.