Classification of Butterfly Species

Thomas Martin Berger
thberger@stanford.edu

Abstract

This project shows the results of identifying butterfly species with computer vision methods applied on a dataset provided by the association "Arbeitsgemeinschaft bayerischer Entomologen e.V" about butterfly species existing in the Alpine region. A convolutional neural network with residual layer serves as a baseline method and is compared with a slightly deeper version, a ResNet50 transfer learning model and the Inception-ResNet-v2 approach.
Cause of present species imbalances, class weights are applied on the training process. Overall Inception-ResNet-v2 delivered the best result (Accuracy over 80% and Macro F1-score over 76% for unweighted an weighted loss), while ResNet50 performance was close behind this results.

1. Introduction

The extinction of various species has gained sad notoriety in recent years. Due to human impact on the environment amphibians (40%), reef building corals (32%) and insects (10%) are threatened by extinction, as described by Brondizio *et al.* [6]. To determine if there is a current decline regarding a specific species, scientists are tracking the population size of various animal species.

The data for this project is provided by the association "Arbeitsgemeinschaft bayerischer Entomologen e.V.". This association investigates the butterfly population in the Alpine region. They take pictures of butterflies and upload them on the website (https://www.tagfalterbayern.de/). On this website, different images of butterflies and their photo location is stored in a database. The overall objective includes documenting the butterfly population over time, which is particularly important given the current decline in insect populations. Images taken by members of the association clearly improves the general level of knowledge about the butterfly population in the Alpine region and a "red list" is indicating which butterfly species are at risk, threatened or endangered by extinction. Classifying butterflies species correctly is an important task for scientist working in this environment. An automatic classifier that identifies the correct species of an butterfly can be a very useful tool in order to facilitate the work of the scientists. Furthermore, a well performing classification algorithm could be a useful tool to identify the uploaded image immediately and may encourage hobby scientist to collect more images from various butterfly species in the nature.

The input to the algorithm are butterfly images of different species. Then different image classification models like ResNet50 are used to identify the butterfly species. Since there are 110 species in the dataset, it is a multi-class classification task.

2. Related Work

Although there are many publications on image classification, not many scientific papers deal directly with applying image classification on butterfly species.

Li & Xiong [22] used Histograms of multi-scale curvature (HoMSC) and gray-level co-occurrence matrix of image blocks (GLCMoIB) to generate features and applied a weight-based k-nearest neighbor classifier to identify the butterfly species.

In Kaya & Kayci [19] artificial neural networks were used to classify butterfly species. Texture and color features were created to achieve a high accuracy. Furthermore, in Kaya *et al.* [20] Gabor- Filter-Based texture features were used to classify butterfly species and Wang *et al.* [35] addressed the problem of identification of butterfly species by using content-based image retrieval. Kang *et al.* [18] considered branch length similarity (BLS) entropy profile as an input feature in a feed-forward neural network to identify butterfly species according to their shapes when viewed from different angles. They achieved the impressive result of $> 90\%$ accuracy for most of the classes. However, they only considered 15 different species and the amount of training data was comparatively low. Although these papers are interesting regarding feature engineering applied on images, the models in this project should learn the features directly from the image data as well as using pretrained models in order to utilize their already existing feature representation.
In recent years other deep learning algorithms like convolutional neural networks (CNN) were applied on image data, because they are better suited for image classification (pre-

serve 2D structure) than fully connected neural networks to capture the feature representation of an image. Arzar *et al.* [5] used a convolutional neural network to determine butterfly species, and achieved great results, but their dataset was very small. Zhu and Spachos [37] successfully utilized a VGG19 (as described in [31]) model to identify butterfly species for a mobile application, but their database was also not very large (< 1000 images). As described by Almryad and Kutucu [4], identifying butterfly species is a difficult task due to huge amount of different species and high similarities between them. In this paper, the authors tested different models like CNNs and transformer models like ResNet to classify butterfly images. However, they considered only 10 different butterfly species, whereas this project covers 110 different species. Furthermore Lin *et al.* [26] applied skip-connections convolutional neural network (S-CCNN) on a butterfly dataset and reached high accuracy. However, their dataset considered only high-fidelity lab-made images captured under the stable illumination conditions of hollow LED shadow less lamp. In contrast to their data, the images used for this project consists of images taken in the nature. Hence, there is noise (e.g. flowers and fields) in the images.

As stated in van Horn *et al.* [34], classification of species can be very difficult, since their diversity is not uniformly distributed. Instead, nature is very imbalanced and some animals are easier to photograph. This is also true *within* a group of insects, e.g. butterflies. Some butterfly species are more common than others, both in nature and in pictures.

3. Data

In the final prepared dataset are 89169 images of butterflies. One dataset is provided by the association "Arbeitsgemeinschaft bayerischer Entomologen e.V." and also some images from a kaggle dataset were used (https://www.kaggle.com/datasets/gpiosenka/butterfly-images40-species). Overall, there are 110 different butterfly species with different number of training examples. The most frequent species is called *Polygonia c-album* with 3633 observations and the rarest species in the dataset is called *Maculinea rebeli* with only 26 observations. Table 1 shows a sorted extract of all classes, with 10 classes accounting for about 25% of the total training examples.

There are some important facts about butterfly appearance especially with respect to image classification.

Within a species (e.g. Maculinea rebeli) the butterflies can be different regarding their lower side (left image in Figure 1) and upper side (right image in Figure 1). Additionally, males and females are often different within a species.

While the butterflies in Figure 1 are clearly recognizable, there are also other images, where the butterfly is not easy to detect. An example is shown in Figure 2

Species name	Number of observations
Polygonia c-album	3633
Gonepteryx rhamni	3096
Aglais urticae	2487
Anthocharis cardamines	2346
Iphiclides podalirius	2273
Argynnis paphia	2210
...	...
Euphydryas cynthia	258
Boloria thore	199
Oeneis glacialis	119
Plebejus argyrognomon	47
Maculinea telejus	45
Maculinea rebeli	26

Table 1. Overview of species name and their corresponding occurrences in the dataset

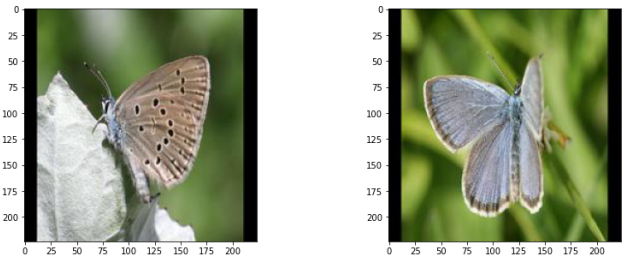

Figure 1. Maculinea rebeli examples

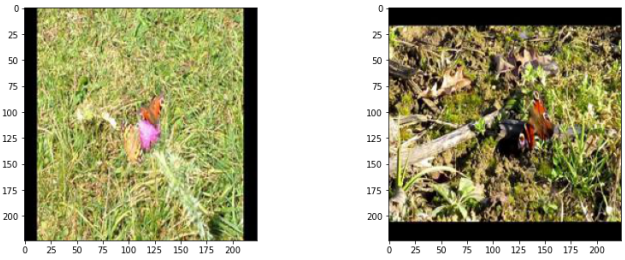

Figure 2. Aglais Io butterfly examples

This, the high number of different species, their general similarity and especially the low number of observations for some classes makes this task challenging.

As stated in 4.4, training data is increased by rotating and flipping and other methods. This is completely reasonable, since the butterflies are photographed from different angels. As mentioned in section 2 on related work, this is in contrast to other work that includes only images of the front of the butterflies. As an image preperation step, all images were resized to 224x224 in order to have a common input shape for the models.

4. Methods

4.1. Convolutional Neural Networks

Convolutional Neural Networks are known as an integral component with respect to image classification. In contrast to plain neural networks, CNNs do not have to stretch the image input, but preserve the spatial structure as stated by Li *et al.* [23]. An example of a convolutional layer is shown below:

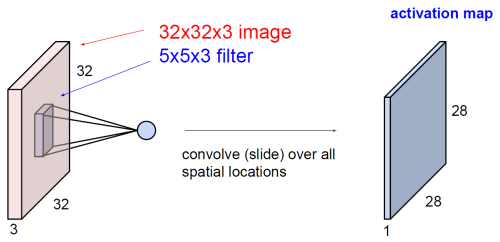

Figure 3. Convolutional Layer, as described in Li *et al.* [23]

As described in Goodfellow *et al.* [10], one of convolutional neural network's (CNN) main characteristic are sparse interactions. These is accomplished by making kernel smaller than the input. Hence, when processing an image with several thousand pixels, only meaningful features like edges are preprocessed. This leads to a sparse model that requires less memory and is statistically more efficient.

Additionally, so called *pooling* layers can be added within convolutional neural networks. E.g. Max-pooling deals with overfitting by only processing the max output of e.g. a 2x2 filter and makes the model more invariant to small translations of the input, as stated in Goodfellow *et al.* [10].

A main layer block for all models in this project is a convolutional layer followed by batch normalization (described in Ioffe and Szegedy [17]) and a ReLU activation function (as stated in Goodfellow *et al.*). By scaling the activation layers, gradient flow is improved, which makes networks learning much more easier. Additionally, it allows higher learning rates which leads to faster convergence and the neural network becomes more robust to unfavorable weight initialization, as stated by Li *et al.* [24].

Furthermore, the softmax function $\frac{\exp(z_k)}{\sum_k \exp(z_k)}$ was considered as output activation function for the respective last layers, as described in Goodfellow *et al*. With this function, an input image is given a score between zero and one, and the image is assigned to the species with the highest score.

4.2. Residual Learning

Residual Learning was a key element of successfully applying deep learning to classify butterfly species. Hence, this subsection takes a closer look at this topic.

From a naive point of view one could think, that adding more layers just making the feature representation better, which should lead to an overall better performance. However, as stated by He *et al.* [13], the *degeneration* issue leads to the fact that deeper models can be worse in terms of training and test error than a suitable model with not so many layers. The degeneration issue is observed when adding layers to a well working model, when training accuracy is saturated and starting to drop. As described by He *et al.* [13], this is not due to overfitting, but because deeper models are more difficult to optimize than their shallower representations. He *et al.* [13] showed this behaviour by adding identity mapping layers to a well working model. The idea is, that this should not lead to higher training errors, but this was actually the case. To solve this issue the *deep residual framework* (as shown in figure 4) was created, which includes the concept of *shortcut connections*. These shortcut

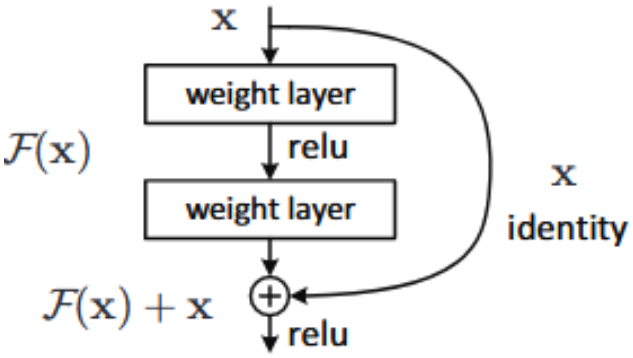

Figure 4. Residual learning, a building block as described in He *et al.* [13]

connections perform identity mappings as it seems easier to optimize the residual mapping than to optimize the original mapping as described in He *et al.* [13]. These mappings ensure that the deeper layers perform as least as good as the lower layers and diminish vanishing gradient issue. A classical combination of layers includes a "weight layer" (mostly convolutional layer) followed by Batch Normalization and a ReLU activation function. After two of such layers an identity connection is added as shown in Figure 4.

4.3. Inception Layer

Instead of considering a convolutional layer with filter size five or three or applying a pooling layer, multiple convolutions with multiple filters and pooling layers simultaneously in parallel within the same layer can be applied. This is called inception layer, as shown in Figure 5

In contrast to a basic convolutional layer, applying multiple convolutional layers let the network learn from multiple layer at one time so that the next stage can abstract features from different scale simultaneously, as described by Szegedy *et al.* [33]. In general, the left architecture in Figure 5 is computationally too expensive, because merging

(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 5. Inception architectures, as described by Szegedy *et al.* [33]

the output of the pooling layer with the other convolutional layers would increase the number of outputs from stage to stage, as stated by Szegedy *et al.* [33].
Thus, the right architecture in Figure 5 was designed. As described by Szegedy *et al.* [33], 1x1 convolutional layers are used before applying 5x5 or 3x3 layers to save computation by reducing the number of channels in the beginning and apply parallel computation.

4.4. Data Augmentation

As stated in Goodfellow *et al.* [10], one way to get a more generalized model is to consider *Data Augmentation*. Generally, e.g. rotating or shifting an existing image is understood as data augmentation. Hence, you generate additional images during training of the respective model. Through this generalization, the model is more robust to overfitting and therefore data augmentation is considered for all models in this project. Following data augmentations were used during training:

- Rotating
- shift width
- shift height
- zoom
- horizontal flip

4.5. Loss Functions

4.5.1 Categorical Cross Entropy

Since the dataset consists of 110 different butterfly species, multi class cross entropy

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i)$$

is considered as loss function for the models, as stated in Friedman *et al.* [9]. y_{ik} is one if observation i belongs to class k and is zero otherwise. $f_k(x_i)$ is the probability estimate of the current image x_i belongs to class k. Notice, that every observation in the above loss is equally weighted.

4.5.2 Weighted Categorical Cross Entropy

To account for the class imbalance present in the dataset, weighted categorical cross entropy

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} w_k y_{ik} \log f_k(x_i),$$

is computed as described by Ho & Wookey [14], whereas this loss is exactly equivalent to cross entropy loss, except the weighting factor w_k. This weighting factor equals to one for the species with the most observations and is greater than one for classes with lower observations. Hence, the loss shifts the model to focus more on the classes with fewer observations.

4.6. Performance Metrics

4.6.1 Accuracy

Accuracy is the main metric computed for image classification tasks. As described by Grandini *et al.* [11], it is computed by the correct predictions divided by the number of predictions.

4.6.2 F1-score

Accuracy is not ideal for measuring classifiers performance if the data has an imbalanced class distribution. Therefore F1-score is considered to deal with this issue.

The F1-score

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

for one class, as stated by Grandini *et al.* [11], is the harmonic mean of Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$. As shown in the formula, precision is the fraction of correctly identified e.g. "Maculinea rebeli" butterflies observations divided by the number of observations which are classified as Maculinea rebeli. Staying at this example, Recall would be the fraction of correctly identified Maculinea rebeli examples divided by all actual Maculinea rebeli examples in the dataset.

For each model, the macro F1-score is computed as described by Opitz & Burst [27]. It is the average of all individual F1-scores :

$$\mathrm{F1} = \frac{1}{K}\mathrm{F1}_1 + \frac{1}{K}\mathrm{F1}_2 + \cdots + \frac{1}{K}\mathrm{F1}_K$$

So, the number of observations in a class does not count and every individual F1-score is equal weighted for the resulting macro F1-score. Thus, it is a measure that penalizes a classifier that focuses too much on classes with many observations.

4.7. Considered Models

4.7.1 Baseline Model

The baseline model is similar but not as deep as the models described in Table 1 in He *et al.* [13]. Similar to the table in the paper, the considered layers are shown in Figure 6.

The most important part regarding the baseline is the inclusion of a residual layer. Without the residual layers, there is a massive issue in terms of vanishing gradients. Hence, residual learning is also used in the baseline, to create a solid model for classifying butterfly species. Overall, the baseline consists of seven layers, where every convolution layer is followed by batch normalization and a ReLU activation function. This procedure (batch normalization before ReLU) is suggested by Ioffe and Szegedy [17] and was considered for ResNet as stated in He *et al.* [13] as well as for the baseline. Between the different layer names (conv1, conv2, ...) residual mapping is considered to ensure stable backpropagation. To keep things simple, the baseline has only a residual layer with respect to conv1 and conv2.

Layer name	Output size	Baseline Model		Residual Net 11		ResNet50	
conv1	112x112	7x7, 64, stride 2					
conv2	56x56	3x3, max pool, stride 2					
		3x3, 64 3x3, 64	x1	3x3, 64 3x3, 64	x2	1x1, 64 3x3, 64 1x1, 256	x3
conv3	28x28	3x3, 128 3x3, 128	x1	3x3, 128 3x3, 128	x2	1x1, 128 3x3, 128 1x1, 512	x4
conv4	14x14	3x3, 256 3x3, 256	x1	3x3, 256 3x3, 256	x1	1x1, 256 3x3, 256 1x1, 1024	x6
conv5	7x7	----------				1x1, 512 3x3, 512 1x1, 2048	x3
	1x1	Average pool (dropout[0.5]), 110d fully connected Layer, softmax					

Figure 6. Layer overview for Baseline, Residual-11 and ResNet50

4.7.2 Residual-11

The second model computed on the dataset was an extension to the first baseline. It includes additional convolutional as well as additional residual layers between conv3 and conv4. Similar to the baseline a dropout was added to deal with overfitting.

4.7.3 ResNet50

The ResNet50 model is considered due to its good perfromance on the ImageNet dataset (described in [29]). ResNet50 was introduced in He *et al.* [13] as stated above and it is pretrained on the ImageNet dataset in order to classify images into categories. Based on its big data foundation and its deep residual framework ResNet50 has learned a rich feature representation. In contrast to the baseline model, it has over seven times as many layers and can be viewed as an big extension. Most of the model weights are freezed, with exception of the last $\approx 20\%$. This number was obtained by freezing more and fewer layers and comparing the results. As for the baseline and Residual-11 model, the general architecture is described in Figure 6.

ResNet50 and the other models were built via the *Keras* (as described by Chollet *et al.* [7]) package.

4.7.4 Inception-ResNet-v2

As stated by [32], Inception-ResNet-v2 combines the ideas of residual layers and inception layers. It considers both approaches and is the deepest network used for this project. The general architecture is shown in Figure 7. For this

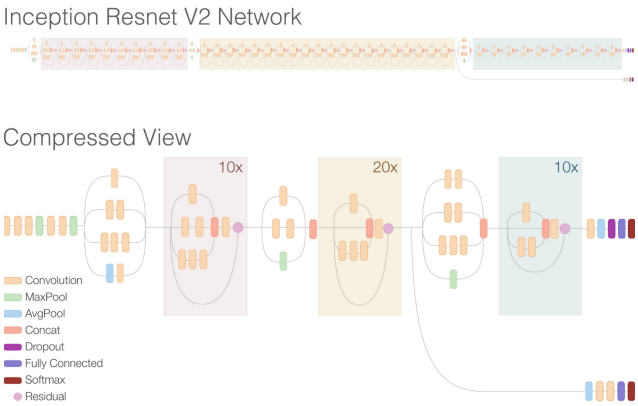

Figure 7. Inception-ResNet-v2 model architecture as described by Ameni [1]

model also the pretrained weights from ImageNet were used to leverage the rich feature representation learned through this big dataset. Just like ResNet50 20% of the layers were unfreezed and retrained with the butterfly data.

5. Experiments

5.1. General Procedure

To train the classifier the images were split into train (80%), validation (10%) and test (10%) dataset to ensure that the classifier is evaluated on unseen data. For all models some general parameters have the same value

- Batch size: 64
- Optimizer: Adam as described by Kingma & Ba [21] with learning rate 0.01
- Maximum number of epochs: 50
- Early stopping if validation F1-score is not increasing: Five epochs

Several other batch sizes and learning rates were tested, but the chosen one delivered the best results. Early stopping was applied to prevent overfitting.

5.2. Class Weights

In order to focus the model's attention on rare classes, class weights were introduced to implicitly introduce a loss function, as stated in subsection 4.5.2. An often suggested formula (as described by Laser [15]) to calculate the respective adjusted class weights w_k is

$$w_k = \frac{\text{\# of observations in most frequent class}}{\text{\# of observations in class } k}$$

As discussed in subsection 5.5, this was not the ideal approach in this project. The formula weights all classes equally, regardless of its original number of observations. Hence, the loss' attention to the class with the lowest number of observations (Maculinea Rebeli with only 20 training observations) is equal to the class with the most observations (Polygonia c-album with 2906 observations). The weight for Maculinea rebeli would be $w_{\text{Maculinea Rebeli}} = 145.3$. This might be a to heavy weight adjustment, because the weights are completely decoupled from their number of observations. Hence, a new formula for the class weights is considered as described by Fernando &Tsokos [8]:

$$w_k = 1 + \log\left(\frac{\text{\# of observations in most frequent class}}{\text{\# of observations in class } k}\right)$$

Now, the overall adjusting is log scaled, which leads to smaller adjustments (e.g. Maculinea Rebeli: $w_{\text{Maculina Rebeli}} = 5.98$). Figure 8 shows the number of observations multiplied with the new weights for some species:

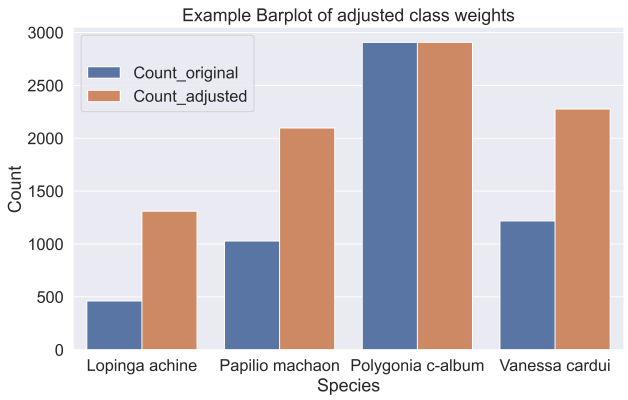

Figure 8. Number of observations scaled with class weights

Notice: with the first formula mentioned in this subsection, all orange bars would be equally high.

5.3. General Results

For all values, final weights of the training iteration with the best validation F1-score were computed in order to receive optimal outcome.

5.3.1 Cross Entropy Loss

Table 2 shows all results without class weighting. Thus, it includes training, validation and test results with respect to accuracy and F1-score:

	Baseline Model	Residual-11	ResNet50	Inception-ResNet-v2
Train Acc	64.22%	69.55%	81.75%	85.07%
Train F1	57.81%	63.19%	76.98%	81.05%
Valid Acc	63.70%	67.52%	79.70%	82.39%
Valid F1	58.13%	62.56%	74.55%	77.57%
Test Acc	64.10%	67.92%	79.82%	**81.93%**
Test F1	57.93%	62.70%	73.77%	**77.56%**

Table 2. Results without class weights

Inception-ResNet-v2 is the best model for the unweighted classification. This is expected, since this model considers pretrained weights and has the most layers combining inception with residual mappings. ResNet50 delivers similar results and is close behind the Inception-ResNet-v2. The Residual-11 model delivers better results than the baseline model. This is also expected, because it has more layers and additional residual layers.

Figure 9 shows the training and validation results for the ResNet50 model. As shown, the curve is saturating over epochs and while training results are nearly continuous improving, validation results are not.

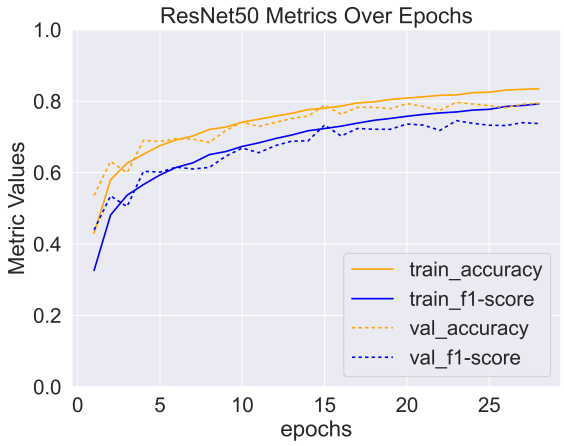

Figure 9. ResNet50

5.3.2 Weighted Cross Entropy Loss

The model results with class weights are shown in table 3.

	Baseline Model	Residual-11	ResNet50	Inception-ResNet-v2
Train Acc	44.58%	78.50%	84.31%	83.51%
Train F1	37.87%	74.76%	81.10%	79.94%
Valid Acc	40.32%	77.13%	80.92%	80.81%
Valid F1	34.44%	74.56%	76.88%	76.44%
Test Acc	40.91%	77.25%	80.92%	**81.35**%
Test F1	34.79%	73.34%	76.88%	**76.94**%

Table 3. Results with class weights

Overall results for all models are improving with exception of the baseline and the Inception-ResNet-v2 model. However, the inception model delivered pretty similar results for both losses. Overall, a better result of the F1-score would be expected, since more weights are adjusted for minority classes. However, the better accuracy is a little surprising, at least for Residual-11 and ResNet50. It seems that a greater focus on on minority classes helps to find a better overall classifier. Also, the comparatively good performance of the Residual-11 model is notable.

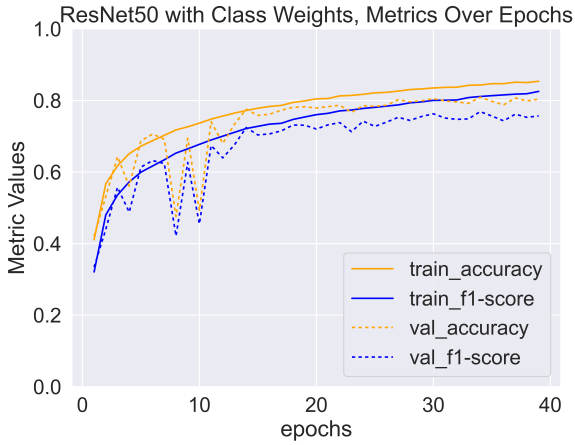

Figure 10. ResNet50

In Figure 10 the training and validation results over epochs are shown. Especially in the beginning the validation results show more variance as in the unweighted case (Figure 9).

5.4. Confusion Matrix and Saliency Maps

The following heatmap and saliency map are generated with the ResNet50 using adjusted class weights.

Figure 11 shows the normalized confusion matrix. The diagonal line shows the **Recall** values for all classes. In general, the diagonal values are lighter than the non-diagonal values. This confirms the overall good model result for the ResNet50 with class weights. However, some diagonal values are very dark and there are some clusters of misclassified values. In the remaining section, an example of a working and a not working example is shown.

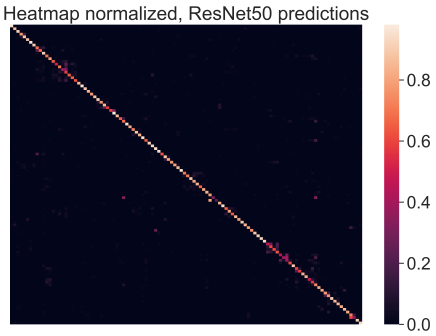

Figure 11. Confusion Matrix

Figure 12 shows an example of a correctly classified butterfly (species: Aglais Io) and the respective saliency map is shown, as stated by Simonyan *et al.* [30].

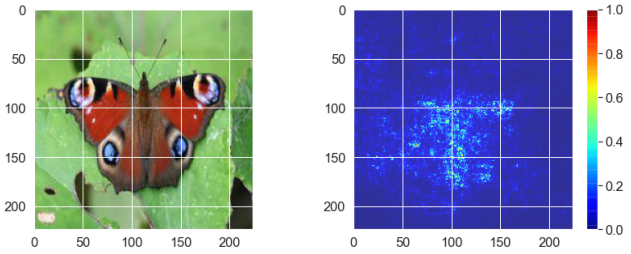

Figure 12. Saliency Map, code adapted from Usman R. [2]

The gradients have a clearly higher value at the location of the butterfly, but there are also minor gradient values which are not located at the butterfly itself.

Many misclassifications are caused by very similar species, as shown in Figure 13.

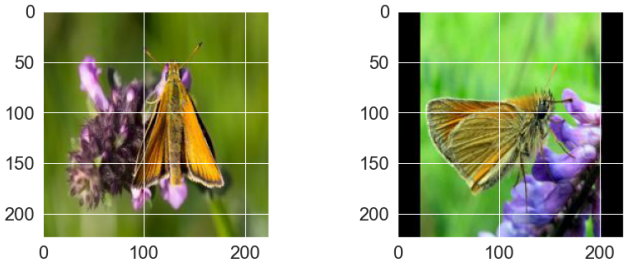

Figure 13. Thymelicus Lineola & Thymelicus Sylvestris, both classfied as Thymelicus Lineola

In this case both species can be distinguished by different colors of the underside of their respective antenna tips: Thymelicus Lineola has a black underside, whereas Thymelicus Sylvestris has a brow-orange underside. This is not easy to recognize at the shown images. This is just one

example of the difficult classification of some species.

5.5. Other Experiments

Focal Loss
As stated by Lin *et al.* [25], focal loss was designed for adjusting cross entropy loss by down-weighting the loss that corresponds to the well-classified classes. Since the dataset of this project also included imbalanced classes, focal loss was also applied on this dataset. However, the loss clearly delivered bad outcomes.

Baseline without Residual Nets
At the beginning, several baselines without residual nets were applied to the dataset. However, most of them fail to classify the baseline adequately. Most of the time the accuracy converged to around 4.1%. This is similar to the fraction of the class with the most observations. As it turned out, the classifier classified all observations to the majority class. This often happens, when the classifier can not derive any useful features from the images and then tries to reduce the loss by classifying all observations into the same class.

VGG16
Introduced by [31], VGG16 is based on a convolutional neural network architecture and won the ILSVR(Imagenet) competition in 2014. Despite its previous success, the results was very poor for this dataset. This is probably due to vanishing gradient problems, because VGG16 does not use batch normalization before ReLU or inherits other vanishing gradient preventing architectures.

Class weights Despite its often mentioned usage on the internet, adjusting class weights for minor classes heavily as described in the first formula in section 5.2 leads to poor results, probably due to overfitting on small number of observations for minor classes.

6. Conclusion

In this project, based on a dataset by the association "Arbeitsgemeinschaft bayerischer Entomologen e.V" butterfly species were classified. This dataset was enriched with some images from the Kaggle website.

Several models were applied and because of imbalanced data, the weighted cross entropy loss was considered. Overall, the Inception-ResNet-v2 model delivered the best results with unweighted loss as well as with weighted loss. Clearly, the usage of pretrained weights and utilizing residual layers was useful for improving accuracy and F1-score.

In future analyses, more models could be tried and more data could be collected for classes with very few observations.

7. Contributions & Acknowledgements

Great thanks are due to the association "Arbeitsgemeinschaft bayerischer Entomologen e.V" for the butterly dataset!

References

[1] Inception-ResNet-v2 architecture description. https://ai.googleblog.com/2016/08/improving-inception-and-image.html. Accessed: 2022-05-31. 5

[2] Saliency Map code. https://usmanr149.github.io/urmlblog/cnn/2020/05/01/Salincy-Maps.html. Accessed: 2022-05-30. 7

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 8

[4] Ayad Saad Almryad and Hakan Kutucu. Automatic identification for field butterflies by convolutional neural networks. *Engineering Science and Technology, an International Journal*, 23(1):189–195, 2020. 2

[5] Nur Nabila Kamaron Arzar, Nurbaity Sabri, Nur Farahin Mohd Johari, Anis Amilah Shari, Mohd Rahmat Mohd Noordin, and Shafaf Ibrahim. Butterfly species identification using convolutional neural network (cnn). In *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 221–224. IEEE, 2019. 2

[6] Eduardo S Brondizio, Josef Settele, Sandra Díaz, and Hien T Ngo. Global assessment report on biodiversity and ecosystem services of the intergovernmental science-policy platform on biodiversity and ecosystem services. 2019. 1

[7] François Chollet et al. Keras. https://keras.io, 2015. 5

[8] K Ruwani M Fernando and Chris P Tsokos. Dynamically weighted balanced loss: class imbalanced learning and confidence calibration of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 6

[9] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017. 4

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 3, 4

[11] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020. 4

[12] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van

Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 8

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 5

[14] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2019. 4

[15] https://datascience.stackexchange.com/users/23478/layser. How to set class weights for imbalanced classes in keras? https://datascience.stackexchange.com/a/13496. Accessed: 2022-05-20. 6

[16] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 8

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3, 5

[18] Seung-Ho Kang, Jung-Hee Cho, and Sang-Hee Lee. Identification of butterfly based on their shapes when viewed from different angles using an artificial neural network. *Journal of Asia-Pacific Entomology*, 17(2):143–149, 2014. 1

[19] Yılmaz Kaya and Lokman Kayci. Application of artificial neural network for automatic detection of butterfly species using color and texture features. *The visual computer*, 30(1):71–79, 2014. 1

[20] Yılmaz Kaya, Lokman Kayci, and Ramazan Tekin. A computer vision system for the automatic identification of butterfly species via gabor-filter-based texture features and extreme learning machine: Gf+ elm. *TEM J*, 2(1):13–20, 2013. 1

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[22] Fan Li and Yin Xiong. Automatic identification of butterfly species based on homsc and glcmoib. *The Visual Computer*, 34(11):1525–1533, 2018. 1

[23] Fei-Fei Li, Jiajun Wu, and Ruohan Gao. Lecture 5: Image classification with cnns, April 2022. 3

[24] Fei-Fei Li, Jiajun Wu, and Ruohan Gao. Lecture 6: Cnn architectures, April 2022. 3

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[26] Zhongqi Lin, Jingdun Jia, Wanlin Gao, and Feng Huang. Fine-grained visual categorization of butterfly specimens at sub-species level via a convolutional neural network with skip-connections. *Neurocomputing*, 384:295–313, 2020. 2

[27] Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019. 4

[28] The pandas development team. pandas-dev/pandas: Pandas, Feb. 2020. 8

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 7

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 8

[32] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 5

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 3, 4

[34] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 2

[35] Jiangning Wang, Liqiang Ji, Aiping Liang, and Decheng Yuan. The identification of butterfly families using content-based image retrieval. *Biosystems engineering*, 111(1):24–32, 2012. 1

[36] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. 8

[37] Lili Zhu and Petros Spachos. Butterfly classification with machine learning methodologies for an android application. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019. 2