Team: ***bisogna_tenersi_idratati_nelle_ore_più_calde***

Image Classification Report

In this document we will initially give a general overview of things we tried and then go over the specific models that we implemented both with transfer learning and without.

1. Trial & Error

Things that actually worked:

- **Oversampling** seemed to give an overall improvement to all models that we tried along the way. The reason we decided to use it (and probably why it worked) is that the given dataset presented quite a bit of unbalance on Species 1 and Species 6. As a result F1 score of both Species 1 and 6 had a noticeable improvement.
- For **early stopping** we tested the difference between monitoring *"val_loss"* and *"val_accuracy"*. We found that the *loss* usually performed a bit better on the public scoreboard. Our guess is that the *accuracy* only accounts for the number of correct predictions while the *loss* also considers how certain that prediction was.
    - On a related note we also tried training the network without early stopping for some epochs to be able to take advantage of the whole dataset (train + validation) but it didn't perform well.
- **Inference-time data augmentation** was another technique that worked quite well for us and was applied to *all the models* we submitted. We simply modified the *model.py* so that our network would give multiple predictions for different rotated versions of the input image. Then we added element-by-element the prediction vectors and used the classic argmax to extract the final prediction.
    - NOTE: at inference time we didn't use the image_data_generator to do augmentation since it would have resulted in non-deterministic predictions. We manually fed the network with deterministic rotated versions of the same image.
- **Model Ensembles** (refer to section 3 - ZUSAMMENLEGEN).
    - Voting
    - Latent feature combination with a dense classifier
- We used **Dynamic Learning Rate** to make sure that when a local minimum is found, we dig deep into it to find the actual minimum value for the loss in that area.
- **GAP** seems to perform better than flattening and results in a smaller number of parameters. All in all it was good even if in this specific case we do not need the flexibility of being able to classify different size images

Things that didn't :(

- To compensate for class imbalance we also tried to assign class weights, giving higher values to Species 1 and 6 but sadly we didn't notice any improvement whatsoever
- Of course we used **training-time data augmentation** to train our model but we noticed that for most models "high" augmentation (Es.width_shift_range=20) yielded very bad results (sometimes models didn't seem to train at all).We think it's due to the fact that the dataset contains *very small images* - 96x96 - and therefore any, even small, loss of pixels due to shifting or rotating *hinders performance.*
    - The solution was to stick with **smaller values of augmentation** (Es.width_shift_range=5)

- **Random cutout augmentation** took a very long time and yielded below-average results. Therefore after few attempts we decided to discard the technique

---

2. Models

As promised here is a commented list of models we tried: first without and then with Transfer Learning

Our Models

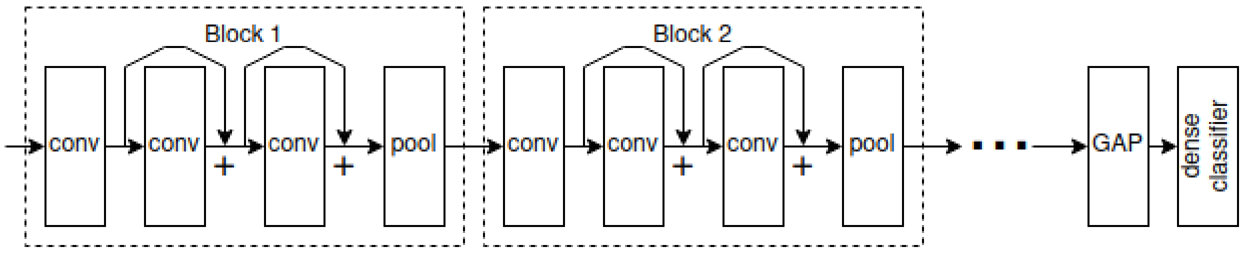
A lot of attempts were made, some more basic, some more elaborate including multiple skip connections and fully convolutional. Here we will give a high-level overview of some of them.

Brief recap of the simpler models (no skip connections):

1. **conv + (conv + relu + maxpool) * + flatten + dense + softmax**
2. **((conv + relu)* + maxpool)* + flatten + dense + softmax**

These models didn't perform particularly well, achieving a public leaderboard *accuracy of around 0.7*

Since attempts without skip connections had very limited performance we decided to move to fancier architectures including skips. Below is a diagram of the best one we found, it achieved a public leaderboard *accuracy of 0.79*



Transfer Learning & Fine Tuning

After the first attempts at building our own networks, where we had hard luck reaching a 0.8 accuracy, we decided to move onto some less creative approach, namely TL & FT.

Most of the models available at Keras Applications were tried. Mentioned below are the most notable attempts.

Before delving into more details let's see how we approached the TL & FT task. We first tried a ***2 step approach***: train only the new network top and then fine tune the whole network with a smaller learning rate. Then we tried a ***1–shot approach*** directly setting to trainable the whole network with a smaller learning rate (but still dynamic like previously mentioned). This second method seemed to work better most of the time.

Now let's discuss models. We started with some smaller networks like MobileNet. The best one of this family was definitely **MobileNetV3Large.** It's the biggest of the family so we figured that more parameters were probably the way to go and so we moved to bigger models.

A really good model family was **EfficientNet**. All the networks with a reasonable number of parameters were tested and they all worked well with different kinds of network tops. The best one was **EfficientnetB3** with GAP followed by Dense top that got a *0.89 accuracy* on codalab

ConvNeXtTiny with a GAP and Dense was the only network that managed to break the *0.9 accuracy* barrier on the 1st codalab phase.

We also tried **VGG16**, **Xception**, **ResNet**, **NASNetMobile**, **InceptionResNetV2** but they all had worse performance compared to the models mentioned above, no matter the top that we combined them with.

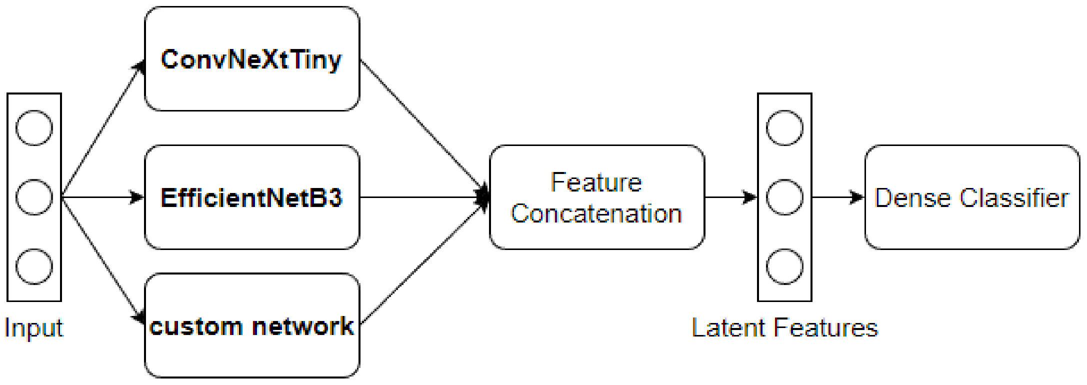
---

3. Model Ensembles

We created model ensembles using voting, latent feature combination with a dense layer and both combined. We present here the best attempts.

ZUSAMMENLEGEN

What is ZUSAMMENLEGEN? Glad you asked!
ZUSAMMENLEGEN is an ensemble of models made with the best networks that we found during the previous experiments. It *combines the latent features extracted by 3 different pre-trained networks* and completes the classification thanks to a fully connected top.



Układ Warszawski (the final model)

Finally, the Układ Warszawski model is an ensemble of:

1. ZUSAMMENLEGEN
2. EfficientNetV2B3
3. ConvNeXtTiny

These networks predictions are combined using a **majority voting** algorithm (predicts the class that gets the most votes). This was our best network in the **2nd phase of the challenge** with an *accuracy on the public leaderboard of 0.898* and therefore **our best overall model**.