Horse Detection and Gait Classification

Anna-Maria von Spreckelsen
Faculty of Computer Science and Engineering
Department Computer Science
Hamburg University of Applied Science
Hamburg, DE

Abstract

Horse detection and gait classification become more important when the online horse competitions will expand. To support the jury in reducing the manual process machine learning might be a good solution. This paper focuses on the horse detection and gait classification. The best results base on a combination of two convolutional neural networks, the EfficientDet D0 for object detection and the Inceptionv3 for gait classification.

Keywords: Convolutional Neural Networks, Object Detection, Classification, Image Processing, Horse Detection, Gait Classification

1. Introduction

More and more horse riding competitions move to online tournaments. For evaluation, competitors have to upload a video of their ride on a website. After that the jury evaluates the video entries to identify the winners. The evaluation is a manual process which can be very time consuming. In order to reduce the manual efforts, this paper evaluates a machine learning approach. Another advantage is a fair, unbiased and consistent ruling.

In this paper the motivation is to extract features like the horse gait. The goal is to identify a setup, based on convolutional neural networks, that is able to detect horses reliable and classify their gait from images.

2. Basics

2.1 Horse's Gait

Many horses have three basic gaits: walk, jog and canter. Some horses e.g. Icelandic horses have two more: tolt and pace. This paper is focusing on the three basic gaits.

Those are visualized in Figure 1. The first row of horses shows the process for walk. Walk is a 4 tact rhythm with 8 phases. The second row shows the process for jog. Jog is a 2 tact rhythm with 4 phases. Finally the third row shows the process for canter. Canter is a 3 tact rhythm with 6 phases. A human eye can see that the tail, the legs, the back and the neck are indicators for each gait.

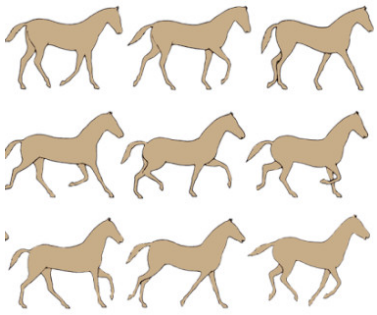


Figure 1: Horse gait process
First row: walk
Second row: jog
Third Row: canter

3. Related Work

According to this papers goal another work, [2] analyzed the horse gait, too. In [2] the horse gait based a body mounted sensor technology. The sensor data were classified by a fully connected artificial neural network, they achieved a accuracy of 97%.

4. Methods

4.1 Dataset

The collected data are part of videos recorded by different smartphones. Depending on the model of smartphone, all videos have different properties e.g. resolution, brightness and background etc.. All videos were taken during horse competitions, dressage, at different areas. So every video has a different background e.g. cornfields, forests or halls. Another background feature is the ground where the horse is competing on. This could be sand or grass, the data include both grounds. Mostly two horse and rider pairs compete against each other. This paper is focusing on the same horse across all videos, the redroan horse, shown in Figure 2.

Figure 2: Redroan horse

Furthermore the videos were cut in subsequences containing only one gait: walk, jog or canter. These gaits are summarized in a set of classes: walk, jog and canter. Every single frame of each subsequence was extracted. After that the redroan horse was marked

with bounding boxes and cropped out later and got a label form the class set. The data distribution for each class is shown in Table 1.

Walk	**Jog**	**Canter**
762	761	785

Table 1: Data distribution for each class

The extracted data were split up in three kinds of data: training, validation and test data. The amount for each kind is shown in Table 2 . In Table 3 the size of training data doubled because the data were augmented via flip-left-right.

Training	**Validation**	**Test**
1544	392	392

Table 2: Data kind amount without data augmentation

Training	**Validation**	**Test**
3088	392	392

Table 3: Data kind amount with data augmentation

4.2 System Setup

To get an overview about the whole system, starting with the dataset, continuing with object detection over to classification, shown in Figure 3.

First raw images, the original images, not edited, serve as input for the object detector. The object detector is a typical pretrained convolutional neural network for object detection. In this case two networks, the SSD MobileNet v2 [4] 320x320 and the EfficientDet [5] D0 512x512 were compared with each other. The object detectors task is to detect horses in the image, draw bounding boxes around them and extract their coordinates. Moreover a preparation module crops the horse, based on its bounding boxes, coordinates and serves the cropped image as input for the classifier. The classifier is a non-pretrained convolutional neural network for image classification. Its task is to classify the horse gait. Three networks, MobileNetV2, a Basic Network and InceptionV3 were compared with each other.

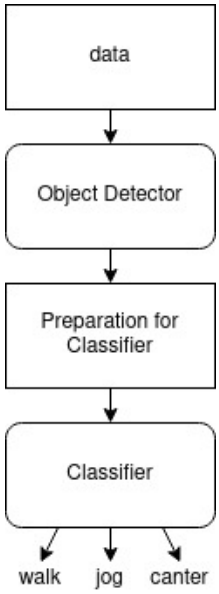


Figure 3: System setup

5. Experiments and Results

5.1 Object Detection

The object detection is an important module in the system. As discussed in subsection 4.2 it detects horses from the input image. First of all two networks, the SSD MobileNet v2 320x320 and the EfficientDet D0 512x512, from the [7] 2 Detection Model Zoo, were chosen for a comparison with the goal to find the best network for horse detection.

At this point five different images per class were chosen from the test dataset and serve as input data for the detector. Next the estimated values of the detected horses were collected and averaged, later saved in a bar chart and compared with each other. Figure 4 gives an overview about the reults for this comparison. Furthermore the graph shows that the accuracy varies across the three classes. The accuracies for jog and canter are lower than the accuracy for walk. An influencing factor is the horse position in the images. The position varies from front to back across the images, shown in Figure 5. The horses which have a smaller size in the image have a lower resolution because all images were cropped and resized to a shape of [299,299]. The SSD MobileNet v2 has continuously lower accuracies as the EfficientDet D0. This indicates that the EcciecientDet D0 is more robust against images with lower resolution as the SSD MobileNet v2. For a better delimitation the focus was extend with some more metrics. In this case speed and mAP value, described in [1].

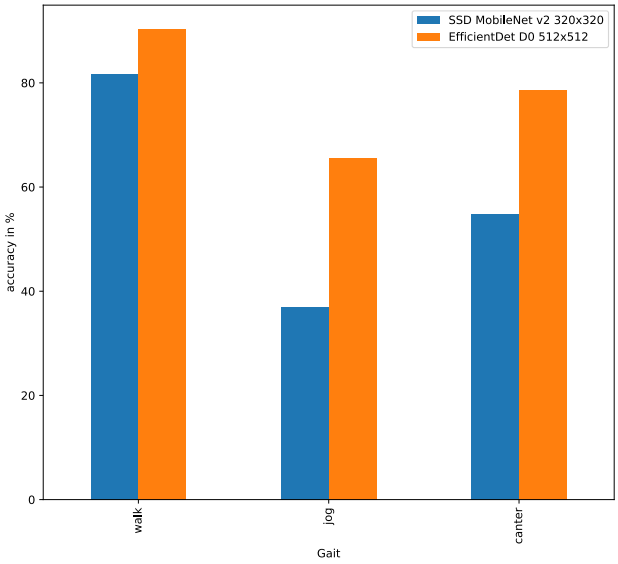
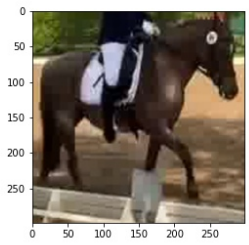


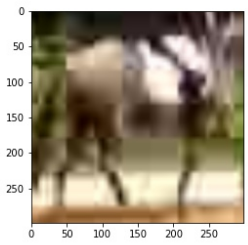
Figure 4: Object detection comparison

(a) Walk original

(b) Jog original



(c) Walk cropped and resized



(d) Jog cropped and resized

Figure 5: Horse position and resolution behavior

The following table includes speed and mAP values for the compared models. Table 4 shows that the SSD MobileNet v2 is 20 ms faster than the EfficientDet D0. This is an advantage for realtime detection. In contrast to this are the mAP values, the EfficientDet D0s value is higher as the SSD MobileNet v2s value. This is indicated by the comparison shown in the bar chart before.

Model	Speed(ms)	mAP Value
SSD MobileNet v2 320x320	19	20.2
EfficientDet D0 512x512	39	33.6

Table 4: Speed and mAP values

5.2 Classification

Another important module is the classification. It classifies the horse gait from given input images. Three non-pretrained neural networks, MobileNetV2, a Basic Network and InceptionV3, were chosen and compared with each other.

At first the architectures were analyzed in more detail. Two networks, the MobileNetV2 and the InceptionV3 are part of the "Keras Application". So their architectures are static and very complex. This classification problem, the horse gait classification, isn't very complex so that a less complex network, the Basic Network, was also chosen. The Basic Network has a similar architecture, shown in Figure 6, like the model in the [6] tutorial for image classification. The differences between them are the missing preprocessing layer and the added dropout layer. The preprocessing for the color rescaling is part of the image preparation and not of the model. The added dropout layer prevents overfitting.

```
model = models.Sequential([
    layers.InputLayer(input_shape=INPUT_SHAPE),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(LABELS))
])
```

Figure 6: Architecture of Basic Network

Before continuing with the comparison of the neural networks, some parameter for model compiling have to be set. This parameters are the optimizer, the loss function and the metrics. Those were set according to the [6] tutorial to the following values shown in Figure 7, valid for all networks. An important parameter for the "Adam" optimizer is the learning rate, default value is set to 0,001, but in this setup it was set to 0,0001. The reason for this modification will be explained later.

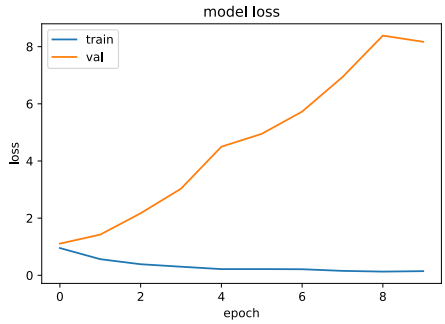
After compilation the model is nearly ready to train, but some steps have to be done before. It's important that the dataset is split in batches and that the epochs, iterations

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
              loss=[tf.keras.losses.CategoricalCrossentropy(from_logits=True)],
              metrics=[tf.keras.metrics.CategoricalAccuracy()])
```

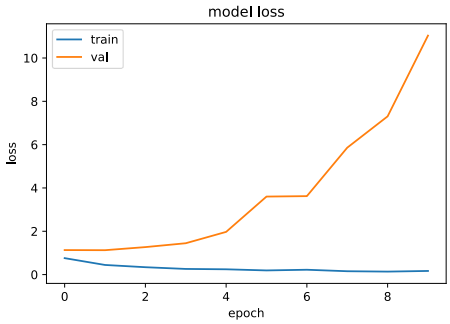
Figure 7: Compilation parameter

that the model needs to train, are set. The batch size was set to 8 and the epochs to 10. The comparison based on different experiments. First loss and accuracy were computed for training and validation per model, based on the first dataset named in subsection 4.2. The learning rate amounts to default, described before. The results of this expermint are shown in the left side graphs from Figure 8 The MobileNetV2 shows a typical overfitting curve, explained in [6], the validation loss is increasing while the training loss is decreasing. This behavior characterizes a model with less robustness. A better model will have two decreasing curves that approach to each other. In contrast to this model the loss curves of the Basic Network are improved, both curves decrease and converge to zero, but it's still overfitting too. Finally the third model, the InceptionV3, has similar curves. The training curve is converging to zero from the beginning, while the validation loss has a big spike in its progress, but later both curves converge to zero. The compared results show that there are overfitting and spikes in the loss progresses. To prevent this, the [6] tutorial offers the option data augmentation.

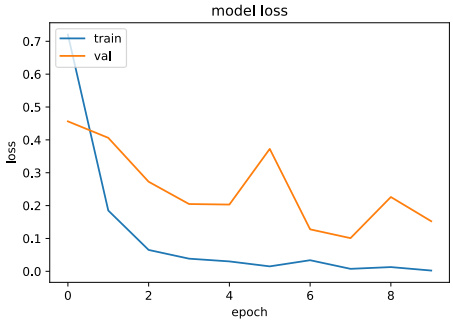
The next two experiments work with the second dataset, shown in Table 3, and default learning rate to prevent overfitting. In this case the graphs on the right side in Figure 8 show the loss progresses with data augmentation. The improvement of data augmentation isn't a benefit at the moment. The validation curves have similar progresses like before. MobileNetV2s curve is increasing and not decreasing, the Basic Network's curve has a higher loss at the beginning and has more spikes than before. Only the InceptionV3 benefits from the data augmentation. It has one spike at the beginning and the loss is a lot lower than before.



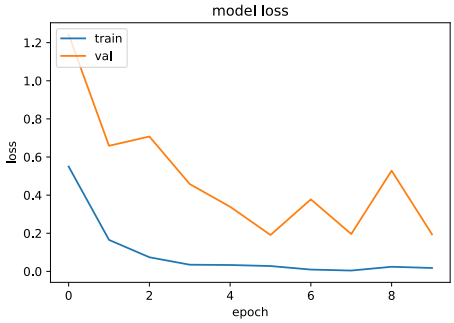
(a) MobileNetV2



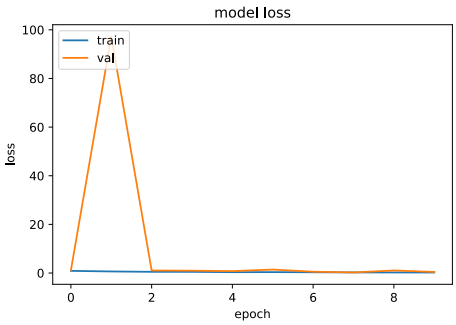
(b) MobileNetV2 with data augmentation



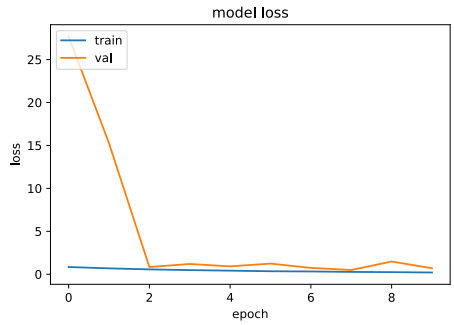
(c) Basic Network



(d) Basic Network with data augmentation



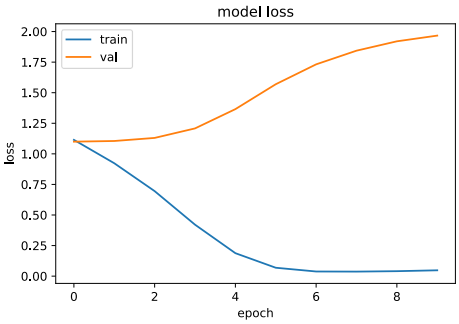
(e) InceptionV3



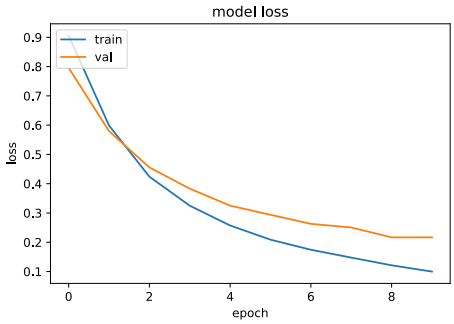
(f) InceptionV3 with data augmentation

Figure 8: Loss comparison

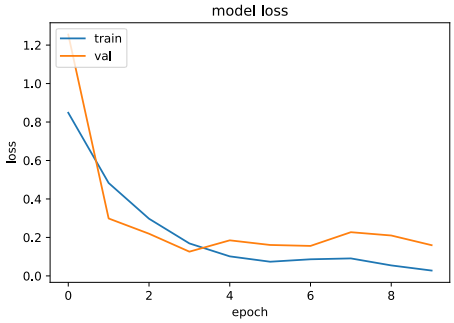
The final experiment based on the previous and but the learning rate is set to 0,00001. The reults of this experiment are shown in Figure 9. In this case the overfitting was decreased across all models. Furthermore the curves of the Basic Network and the InceptionV3 converged to zero. The MobileNetV2s curves drift apart. Finally the Basic Network and the InceptionV3 were chosen for further analysis.



(a) MobileNetV2 with data augmentation



(b) Basic Network with data augmentation



(c) InceptionV3 with data augmentation

Figure 9: Loss comparison with learning rate 0,00001

Next the above evaluated models were tested with a new test set, containing 300 new images which the models had never seen before. The results for accuracy and the average prediction duration were summarized in the following table, Table 5. The results show that

Model	**Test Accuracy in %**	**Average Prediction Duration in ms**
Basic Network	31	11
InceptionV3	85	38

Table 5: Test Accuracy And Average Prediciton Duration

the Inception V3 is the model with best test accuracy of 85%, instead the Basic Network with 31% accuracy. But the Basic Network is much faster in predicting as the InceptionV3.

For getting a better understanding about the InceptionV3 models and its classification the confusion matrix is shown in Figure 10. The confusion matrix shows that the classifier predicts 2 of 100 images for walk as jog, 2 of 100 images of jog as canter and 41 of 100 images of canter as jog. To prevent this behavior, that 41% of canter will be classified as jog, more images for canter might be needed.

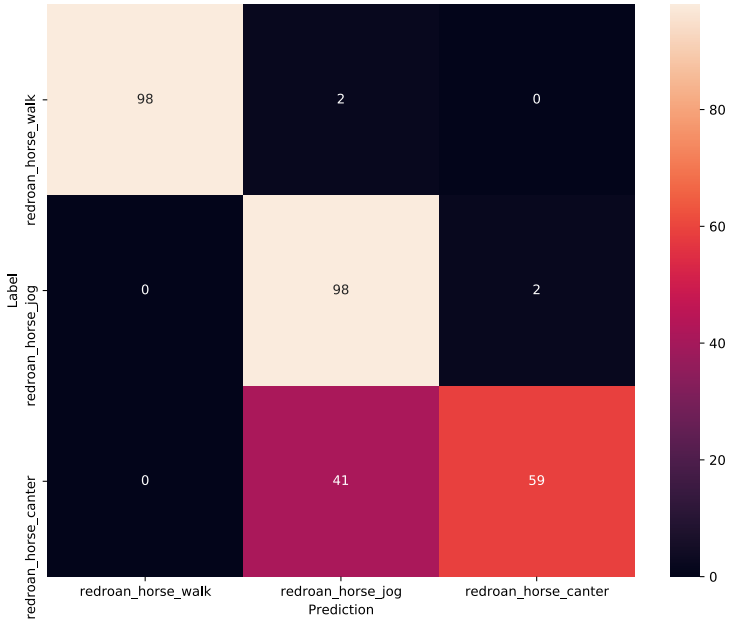
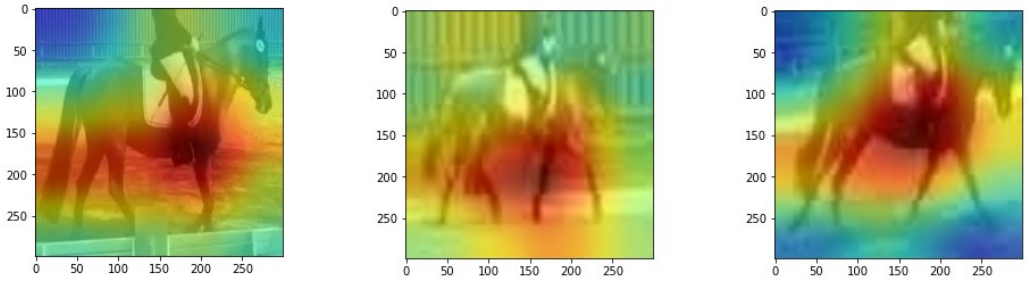


Figure 10: Confusion matrix InceptionV3

Finally it's interesting to see and to know what the classifier learned and what its focusing on. According to [3], heatmaps are fitting instrument for this task. In this case the heatmaps were extracted for all three classes, shown in Figure 11.



(a) Walk (b) Jog (c) Canter

Figure 11: Heatmaps for each gait class

The heatmaps show that the legs of the horse are the most interesting part for the model. The InceptionV3 with the explained setup looks at the same key points like a human would, explained in subsection 2.1.

6. Conclusion

This study has demonstrated that it's possible to detect horses in images and classify their gait.

Two Object Detectors, the SSD MobileNet v2 and the EfficientDet D0 were compared with each other. The comparison shows that the EfficientDet D0 is more robust than the SSD MobileNet v2, but its computing duration is a bit longer. For the papers main goal the robustness is more important than the computing duration because the higher the accuracy of a detected horse, the higher is the accuracy for gait classification.

Furthermore three classifiers, the MobileNetV2, a Basic Network and the InceptionV3, were compared with each other. The comparison shows that the InceptionV3 is the most robust classifier of those three, it's accuracy reachs to 85%. This model has a little longer inference time than the others, but the robustness is more important.

Finally a setup in combination with EfficientDet D0 and InceptionV3 leads to the best results.

7. Prospects

For the future the explained setup has to be ported to the Raspberry Pi 4 Modell B in combination with the Google Coral Stick. This setup has to be analyzed in practice and has to be evaluated in its performance.

References

[1] Steven M. Beitzel, Eric C. Jensen, and Ophir Frieder. *MAP*, pages 1691–1692. Springer US, Boston, MA, 2009.

[2] Filipe Braganca, Sofia Broomé, Marie Rhodin, Sigridur Bjornsdottir, Vikingur Gunnarsson, John Voskamp, Emma Persson-Sjodin, Willem Back, Gabriella Lindgren, Miguel Novoa-Bravo, Christoffer Roepstorff, Berend Jan van der Zwaag, Paul van Weeren, and Elin Hernlund. Improving gait classification in horses by using inertial measurement unit (imu) generated data and machine learning. *Scientific Reports*, 10, 10 2020.

[3] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] TensorFlow. Image classification. `https://www.tensorflow.org/tutorials/images/classification`, 2021. [Online; accessed 20-August-2021].

[7] TensorFlow. TensorFlow 2 Detection Model Zoo. `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md`, 2021. [Online; accessed 20-August-2021].