Team Residuals

Movie Genre Prediction

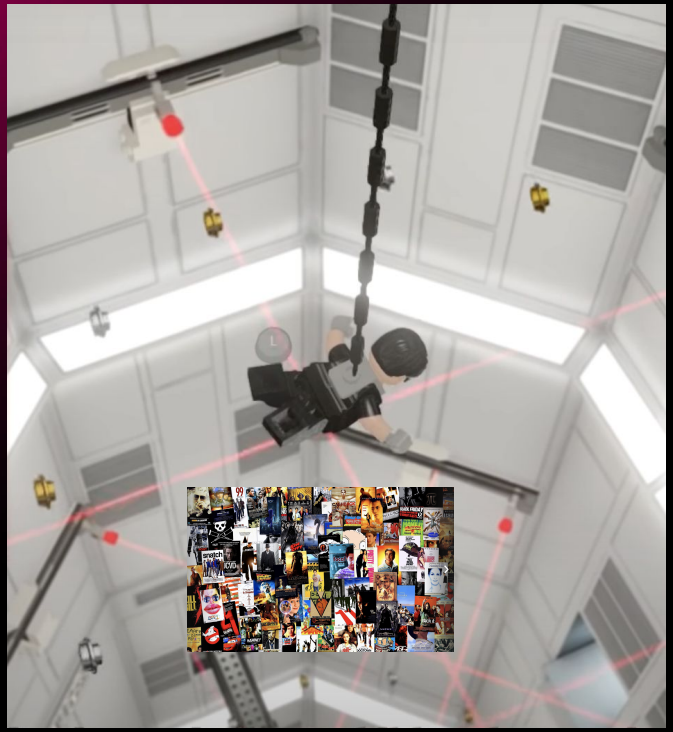
March 24th 2021

Präsentiert von: Andras, Esther, Rainer, Thomas

Agenda

1. Management Overview
2. Technical Details

Mission



Mission: Create one model, which can predict the genre(art) of a given poster.

No matter the:

Size, release date or popularity

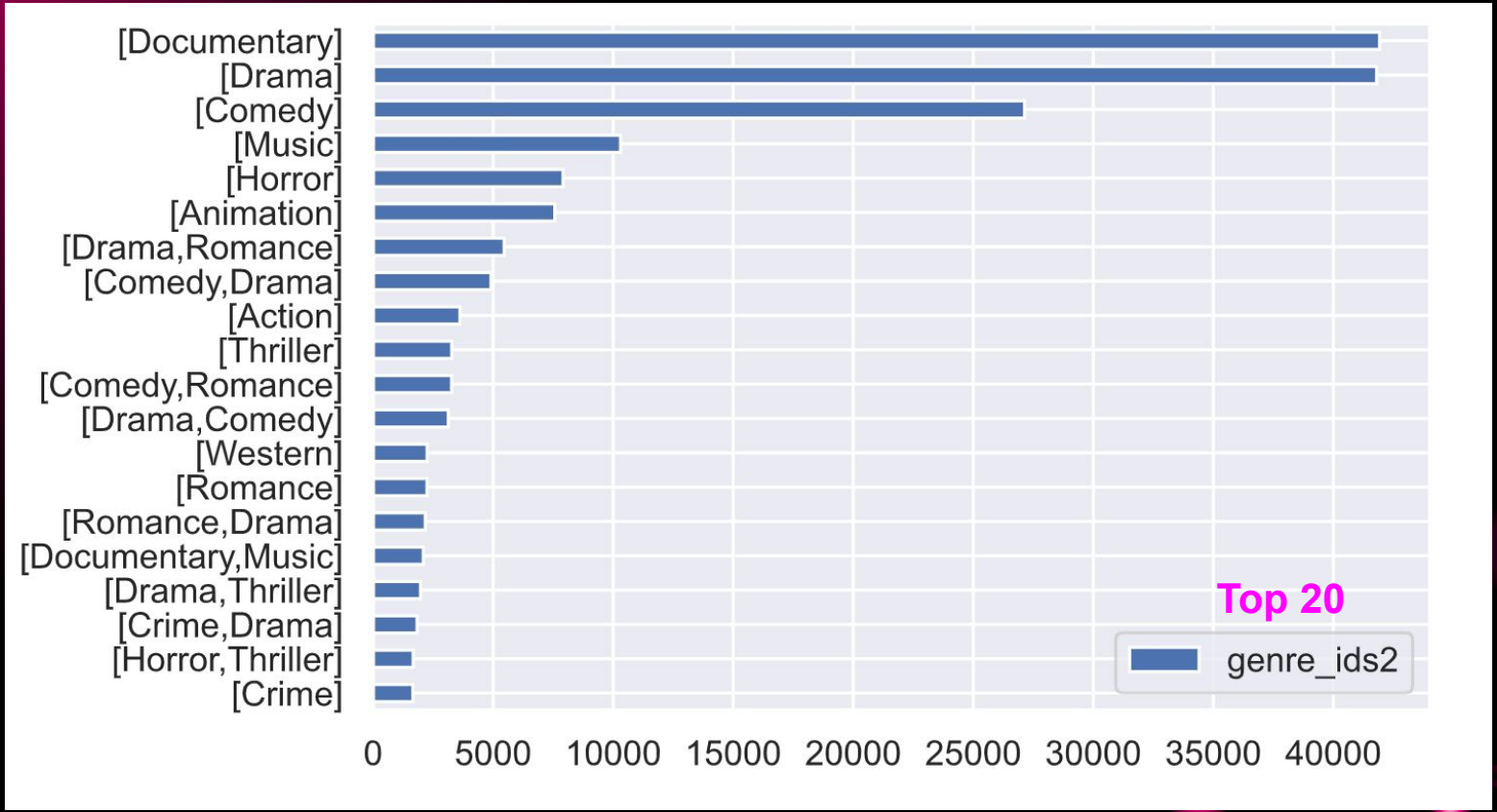
Find it !!!

Our Approach

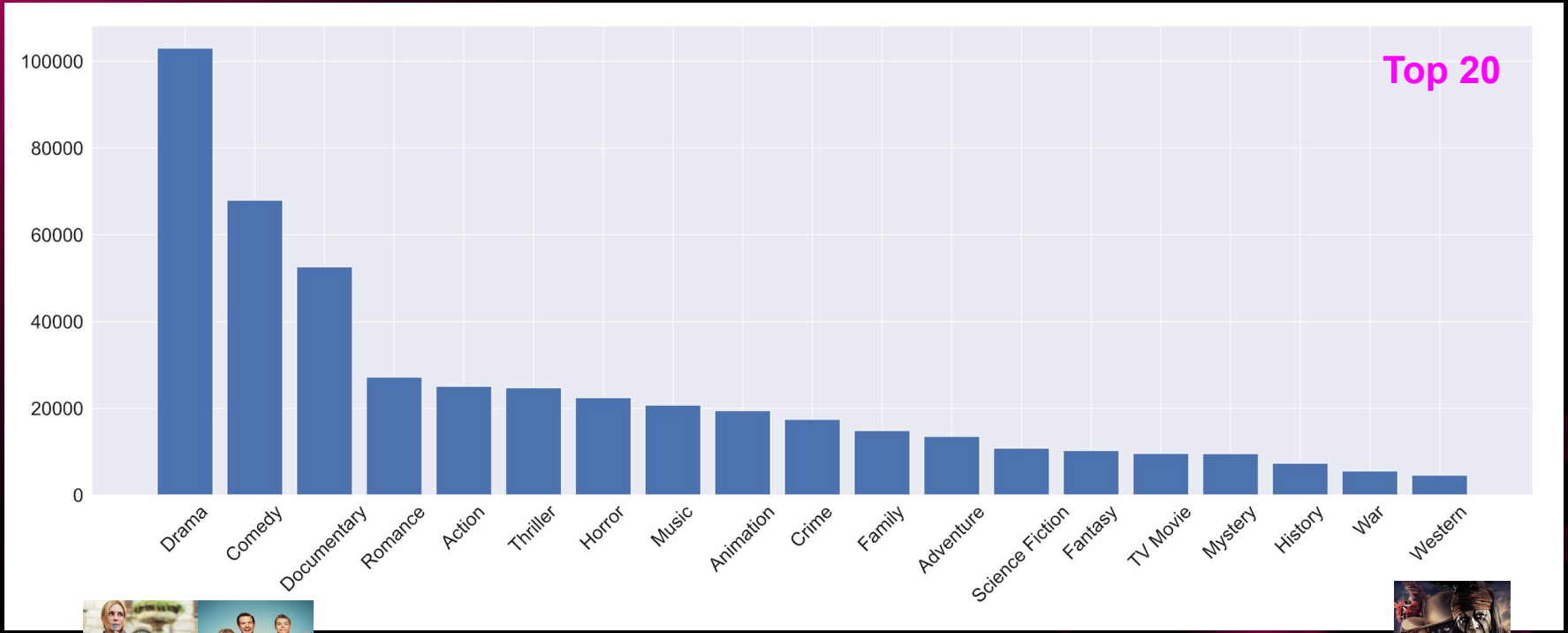
- Multilabel problem understanding
- Unbalanced dataset in the meaning of genres
- Size of the dataset was appropriate for using CNN models

  ~270.000 with valid image posters
- Existing models (pretrained models) or creation of a new one and trained
- We choose metrics that fits for this problem

Multilabel problem understanding.

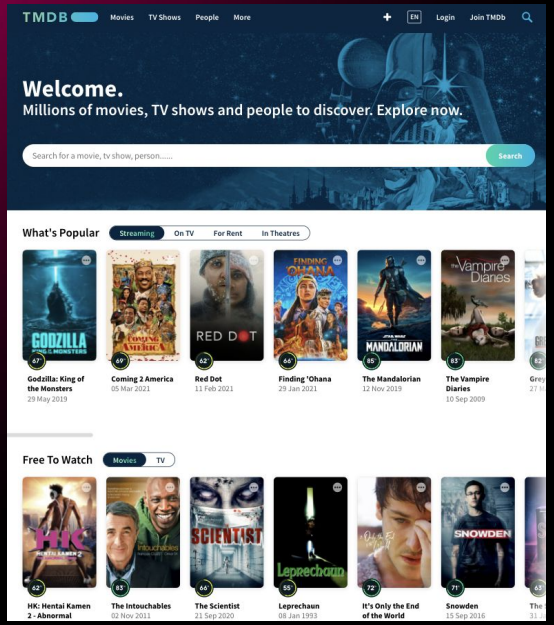


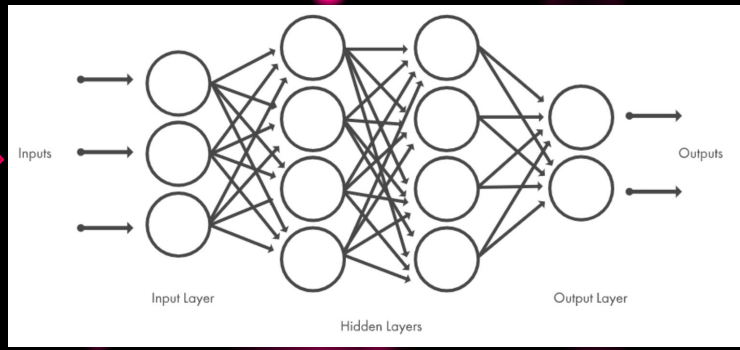
Unbalanced dataset in genres



Size of the dataset

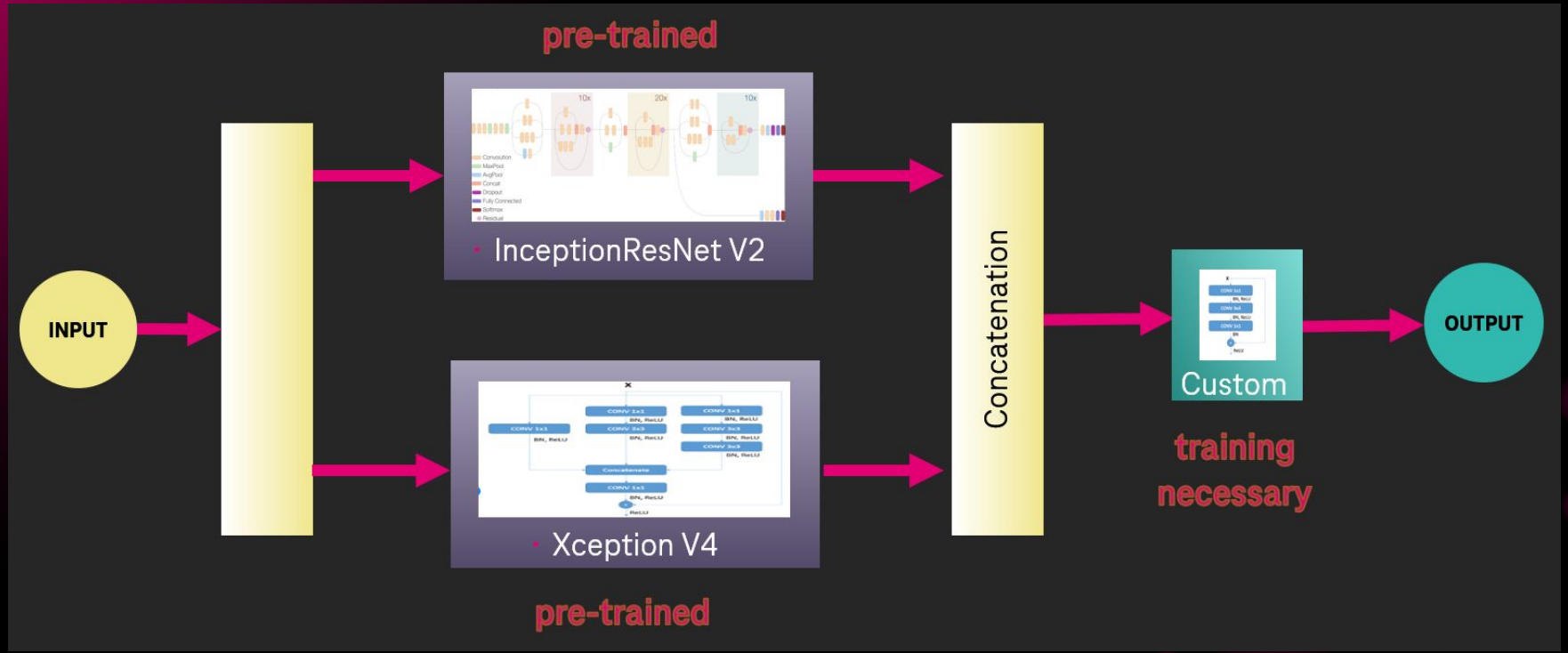
- Aprox ~270.000 entries with valid posters images (21 GB)
- The dataset size was appropriate for using CNN models





Metrics - F1 score validated

Using existing model or build up a new one?



Winner Model

Transfer-Learning Model (Inception ResNet V2.0 + Xception V4), pre-trained

Own Classificator Layers, trained -10 Epochs

Accuracy: 0.434 in test * and 0.514 in train

Training time: 2 Stunden

F1 Weighted Score:0.499

F1 Micro Score 0.516

Reality Check on favorite Movies of the team

Genres TMDb	Action, Science Fiction	Action, Comedy, Science Fiction, Family	Drama, Thriller, Crime, Horror
Predicted	Action, Science Fiction, Thriller	Action, Comedy, Science Fiction, Family	Drama,Thriller

Technical Details

Management of Artefacts

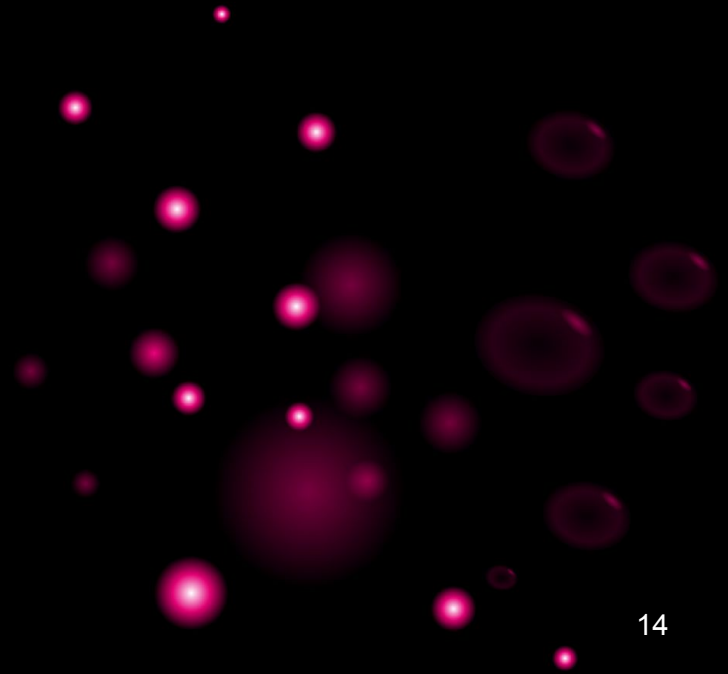
- Git-Hub
  - Source code
  - Small data files (up to 25 MB)

- Google-Drive
  - Movie posters (images) within ZIP archives
  - Presentation

Development

- Python ML and Deep Learning Tools

- Development Environments
  - Google Colab
  - Kaggle Notebooks and Dataframe
  - Local Jupyter Lab
  - Local Visual Studio Code



Preprocessing

- Jupyter Notebooks
- Using downloaded file for looping all movies of Movie-Poster-Database (TMDb)
- Fetching movie metadata using the API of TMDb
- Fetching movie posters (JPG images) via TMDb API
- Storage of resulting data frames for train- and test sets in parquet files
- Storage of images to Google-Drive and Kaggle dataset

Preprocessing

- Cleaned train dataset without posters
  (unbalanced 273,876 rows / balanced 15,015 rows)

- Categorical encoding of genres

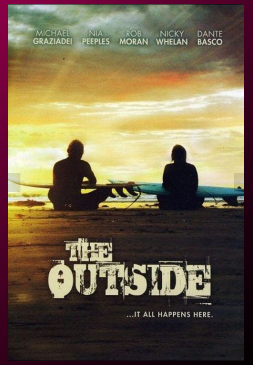
- Testset 1000 rows validated
  Missing images or genre
  Release-dates from 1924 to 2012

Data Quality

- ~500.000 Movies within TMDb
- ~270.000 with valid image posters (21 GB)
- Movie release-dates from 1879 to 2030

Poster Quality

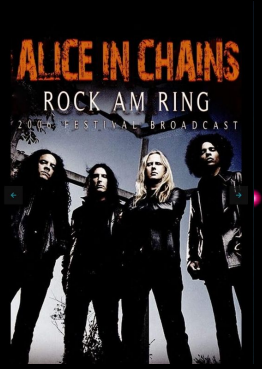
Classic



Old fashioned

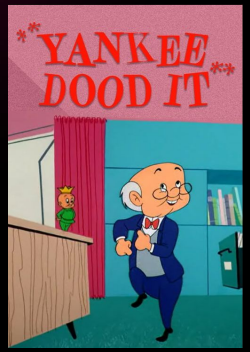
Strange

Music Event



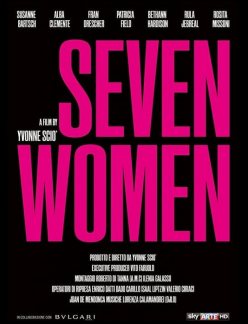
Old fashioned too

Comic



Sports Event

No Picture



Picture only

Picture only

Modeling

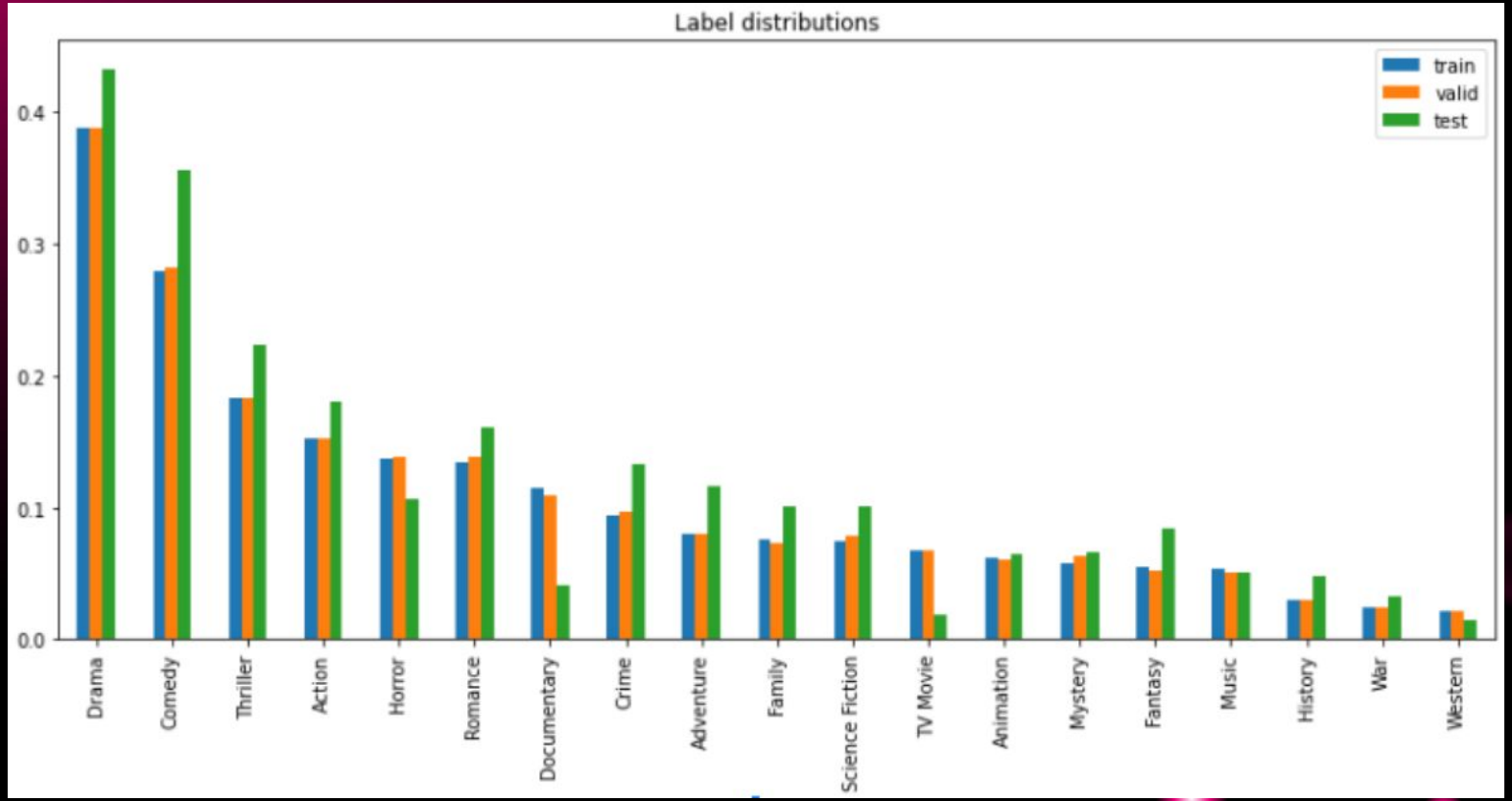
- Keras API
- Keras ImageDataGenerator and Tensorflow Datasets
  Resized images e.g. 299x299x3 for transfer learning
- Performance Issues in Colab and local environments
- Experiment with Kaggle environment und Windows AI Server

Tested Models

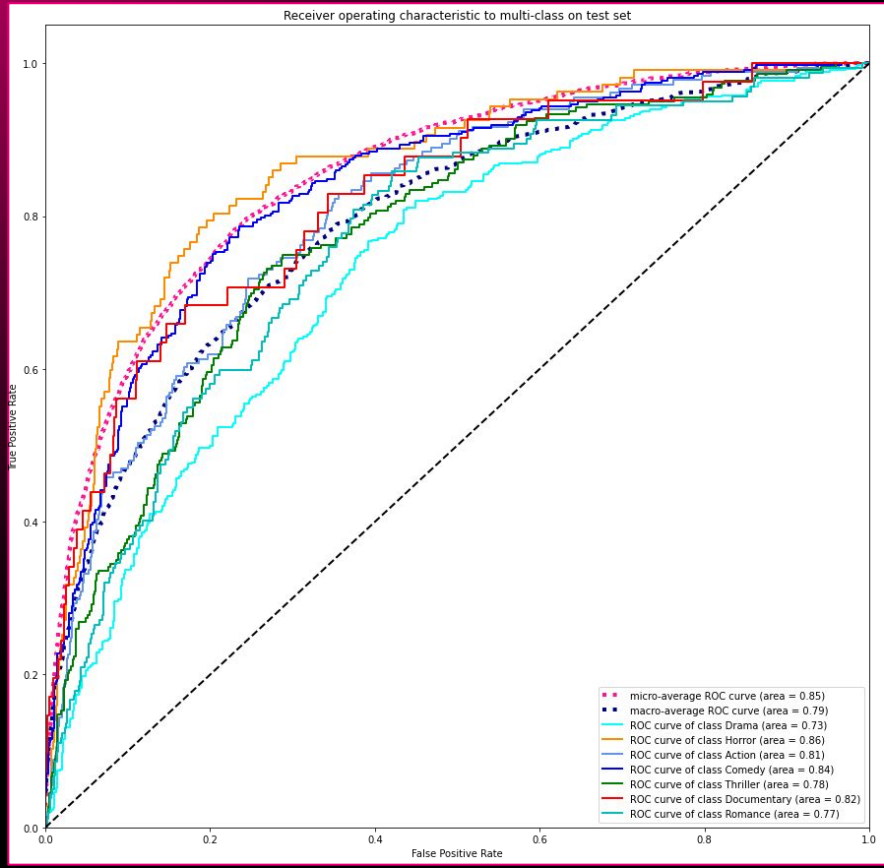
* Threshold = 0.5

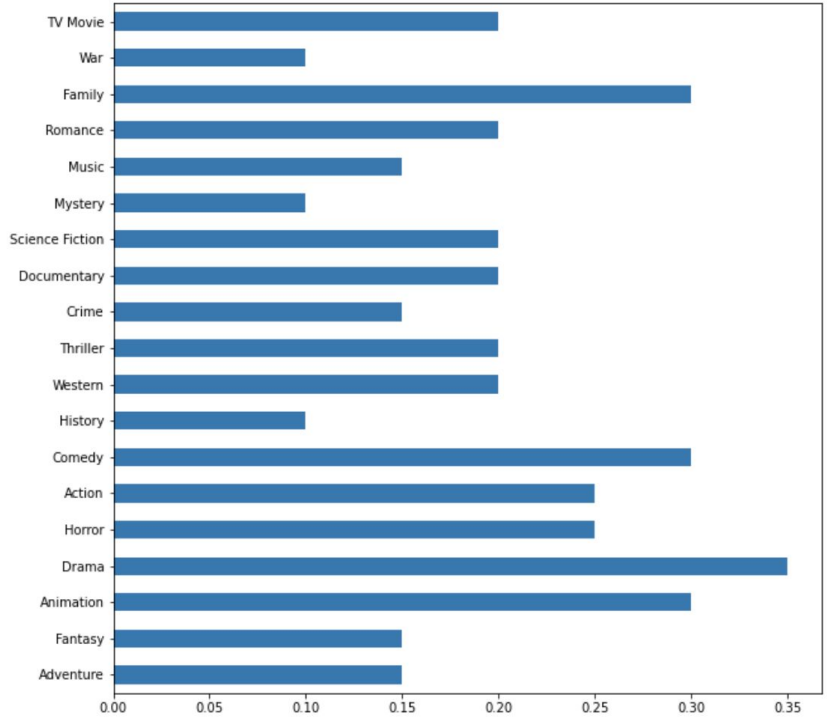
Model	F1 Weighted Score	F1 Micro Score	Training Set
Randomized (Threshold 0.5)	0.079	0.076	-
Simple CNN (baseline)	0.395*	0.346*	~ 266.957
DenseNet (per Class Threshold)	0.317* 0.510	0.363* 0.506	~ 266.957
Inception v3	0.344*	0.382*	~273.876
InceptionResNet V2 + Xception V4 + Custom (Threshold 0.25)	**0.375* 0.499**	**0.417* 0.516**	**~50.000 most popular**

Examine the Poster distribution (unbalanced)



RoC curve and Thresholds (DenseNet)

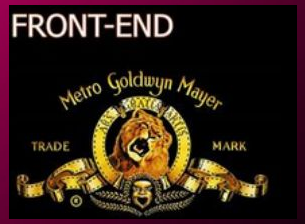




Where we run? / Kaggle

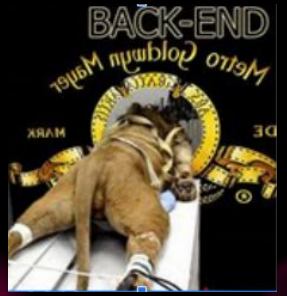
- Kaggle Dataset for Images
- Version in Background
- Private Data 100 GB
- GPU 41 hrs per week
- TPU 30 hrs per week

Where we run? / AI Server



- Remote Desktop

- CPU:
  2 x Intel® Xeon® CPU 55-2690 v4 @ 2.60GHz (28 cores)
  (all in all 56 logical processors)



- RAM: 128 GB

- GPU: 2 x NVIDIA Tesla K80 (11 GB), CUDA 11.2

- Disks: 2 x DEL PERC H730 Mini SCSI Disk Device (149 GB, 1.74 TB)

- OS: Windows Server 2019 Standard 64-bit

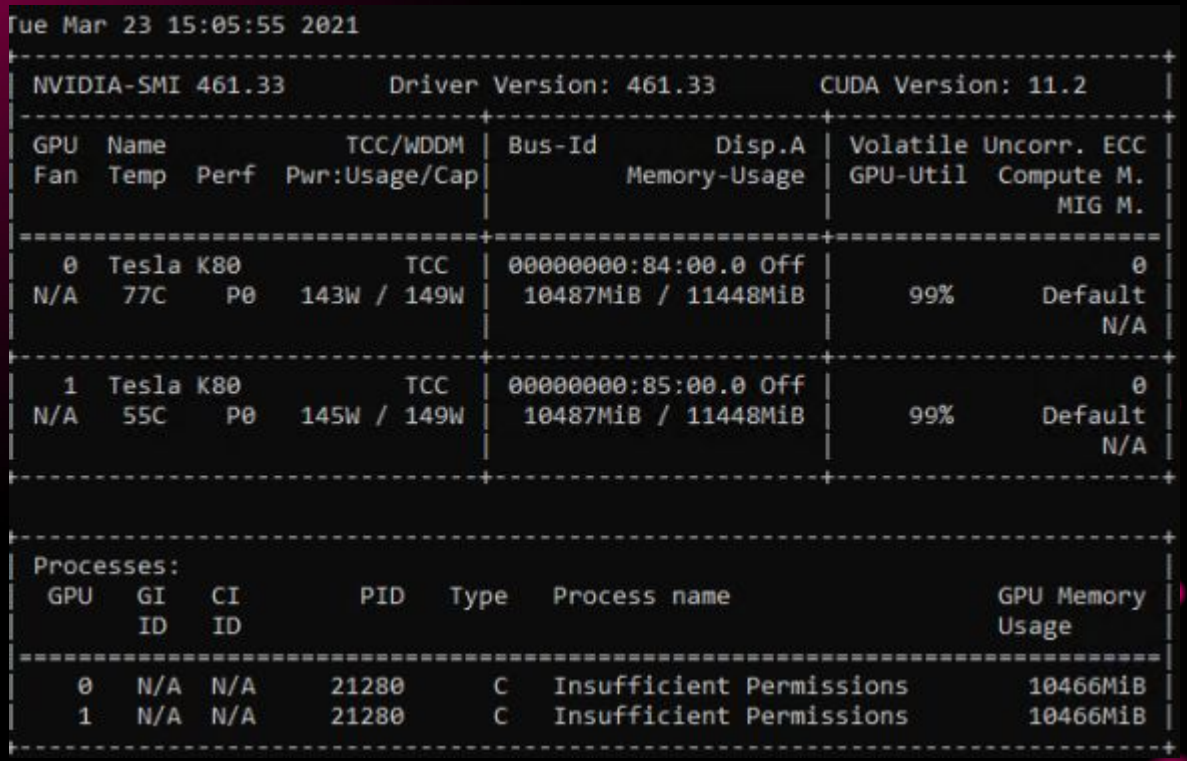
Results Performance Measurements

Test scenario: VGG16 Net, 2 epochs measured, 15k training sets, mirrored strategies

- 2 GPUs are 24 times faster than 56 CPUs
- 2 GPUs are 1.6 times faster than 1 GPU
- 2 GPUs are 1.5 times faster using tf.dataset (cached)
  instead of ImageDataGenerator
  for 1 GPU and CPUs there are no impact measured
- Tensorboard callback does not slow down the training
- NVIDIA Collective Communication Library (NCCL) not working

BACKUP

Usage of 2 GPUs on Win AI Server



2 GPUs using tf.dataset (cached)