Playing Cards image classification

Introduction

In this project, we develop a Mobilenetv2-based model to classify playing cards into different categories. The model is trained on a dataset consisting of images of playing cards from various decks and suits. The goal is to accurately identify the type of playing card depicted in a given image.

Dataset Overview

The dataset used for training and evaluation contains a diverse collection of playing card images. It includes images of cards from different decks, suits, and ranks. The dataset is pre-processed to ensure consistency in image size and format.

- Dataset Size: 7794 images
- Classes: 53 (One class for each type of playing card)
- Image Size: 224 x 224 pixels (RGB format)
- Train-Validation-Test Split: 7624 images / 265 images / 265 images

For more information about the dataset, refer to this link.

What are Dataset Cards?

Each dataset may be documented by the README.md file in the repository. This file is called a **dataset card**, and the Hugging Face Hub will render its contents on the dataset's main page. To inform users about how to responsibly use the data, it's a good idea to include information about any potential biases within the dataset. Generally, dataset cards help users understand the contents of the dataset and give context for how the dataset should be used.

You can also add dataset metadata to your card. The metadata describes important information about a dataset such as its license, language, and size. It also contains tags to help users discover a dataset on the Hub, and data files configuration options. Tags are defined in a YAML metadata section at the top of the README.md file.

Dataset card metadata

A dataset repo will render its README.md as a dataset card. To control how the Hub displays the card, you should create a YAML section in the README file to define some metadata. Start by adding three --- at the top, then include all of the relevant metadata, and close the section with another group of --- like the example below:

The metadata that you add to the dataset card enables certain interactions on the Hub. For example:

- Allow users to filter and discover datasets at https://huggingface.co/datasets.
- If you choose a license using the keywords listed in the right column of this table, the license will be displayed on the dataset page.

When creating a README.md file in a dataset repository on the Hub, use Metadata UI to fill the main metadata:

Uploading datasets

The Hub is home to an extensive collection of community-curated and research datasets. We encourage you to share your dataset to the Hub to help grow the ML community and accelerate progress for everyone. All contributions are welcome; adding a dataset is just a drag and drop away!

Start by creating a Hugging Face Hub account if you don’t have one yet.

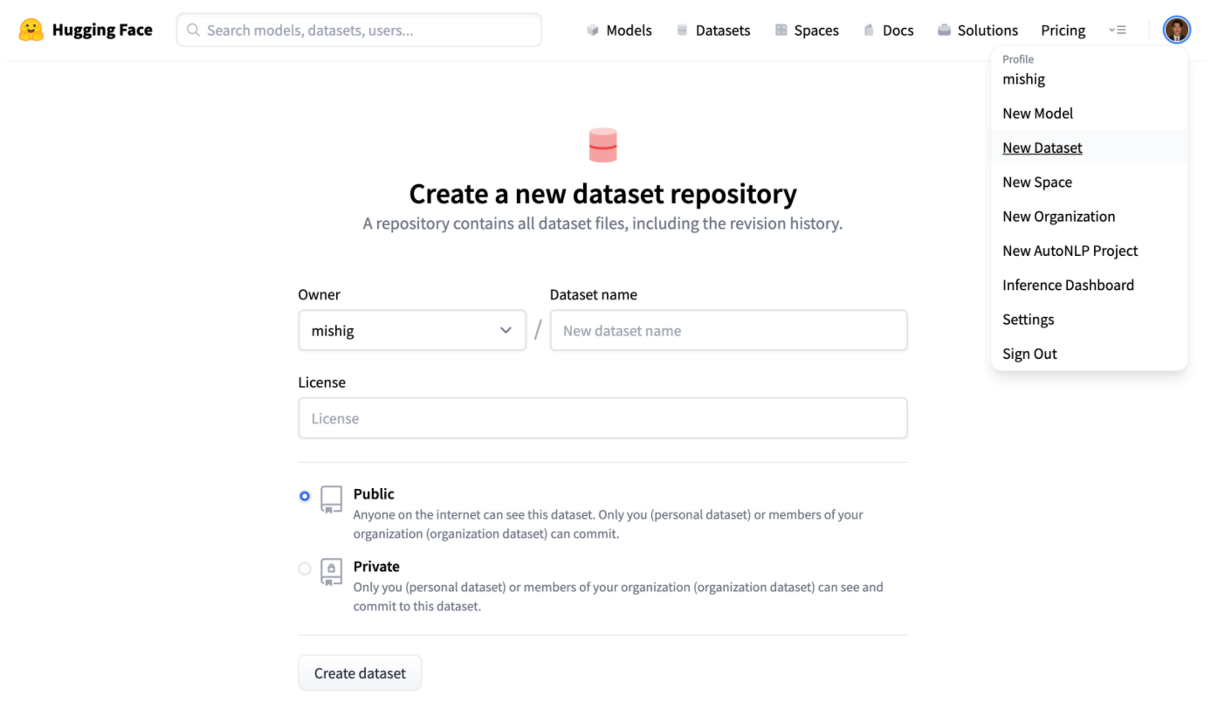
Upload using the Hub UI

The Hub’s web-based interface allows users without any developer experience to upload a dataset.

Create a repository

A repository hosts all your dataset files, including the revision history, making storing more than one dataset version possible.

1. Click on your profile and select **New Dataset** to create a new dataset repository.
2. Pick a name for your dataset, and choose whether it is a public or private dataset. A public dataset is visible to anyone, whereas a private dataset can only be viewed by you or members of your organization.



Create a new dataset repository

A repository contains all dataset files, including the revision history.

Owner: mishig / Dataset name: New dataset name

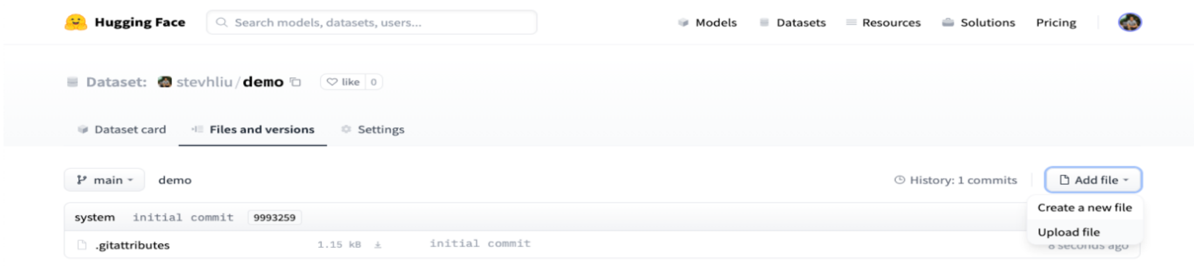
License: License

Public
Anyone on the internet can see this dataset. Only you (personal dataset) or members of your organization (organization dataset) can commit.

Private
Only you (personal dataset) or members of your organization (organization dataset) can see and commit to this dataset.

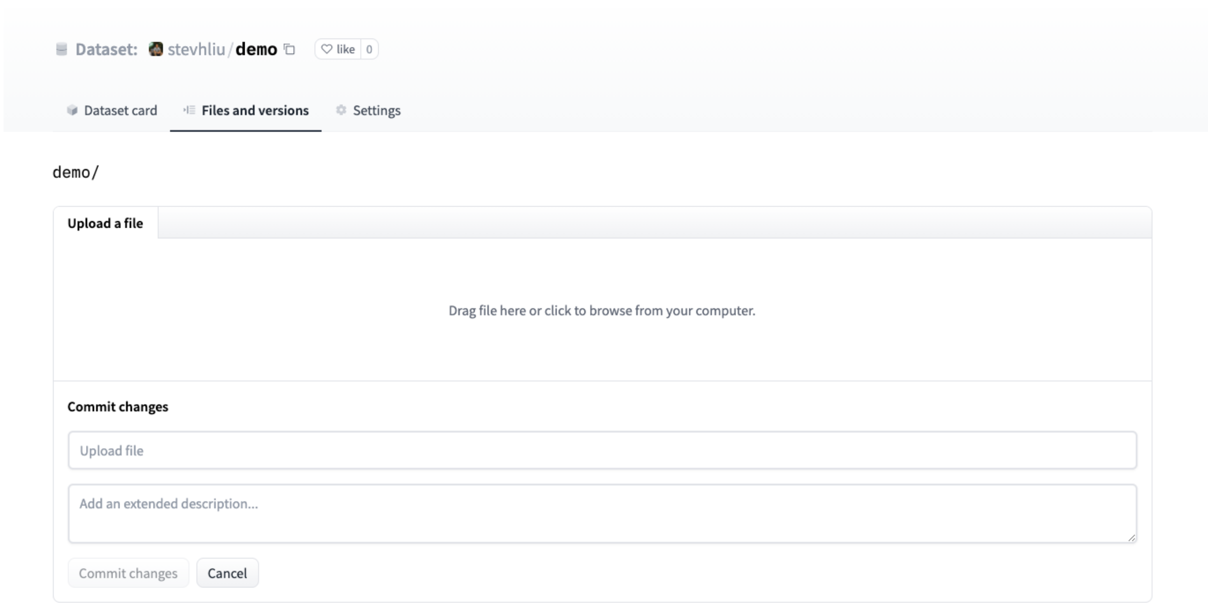
Upload dataset

1. Once you've created a repository, navigate to the **Files and versions** tab to add a file. Select **Add file** to upload your dataset files. We support many text, audio, image and other data extensions such as .csv, .mp3, and .jpg (see the full list of File formats).



```
system  initial commit  9993259
.gitattributes    1.15 kB    initial commit    8 seconds ago
```

2. Drag and drop your dataset files.

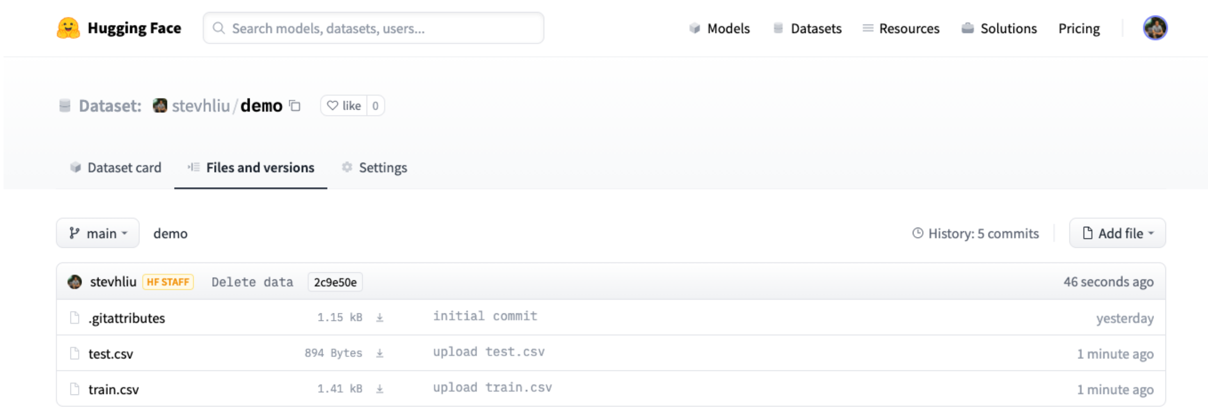

Dataset: stevhliu/demo

demo/

Drag file here or click to browse from your computer.

Upload file
Add an extended description...

3. After uploading your dataset files, they are stored in your dataset repository.



Dataset: stevhliu/demo

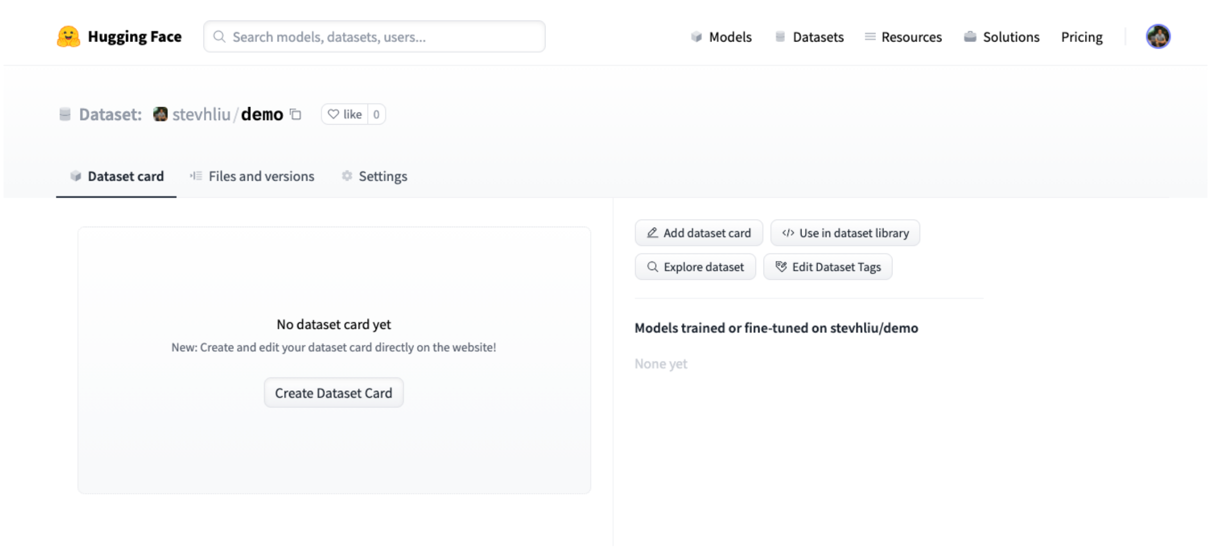
stevhliu HF STAFF Delete data 2c9e50e 46 seconds ago

.gitattributes	1.15 kB	initial commit	yesterday
test.csv	894 Bytes	upload test.csv	1 minute ago
train.csv	1.41 kB	upload train.csv	1 minute ago

Create a Dataset card

Adding a Dataset card is super valuable for helping users find your dataset and understand how to use it responsibly.

1. Click on **Create Dataset Card** to create a Dataset card. This button creates a README.md file in your repository.



No dataset card yet

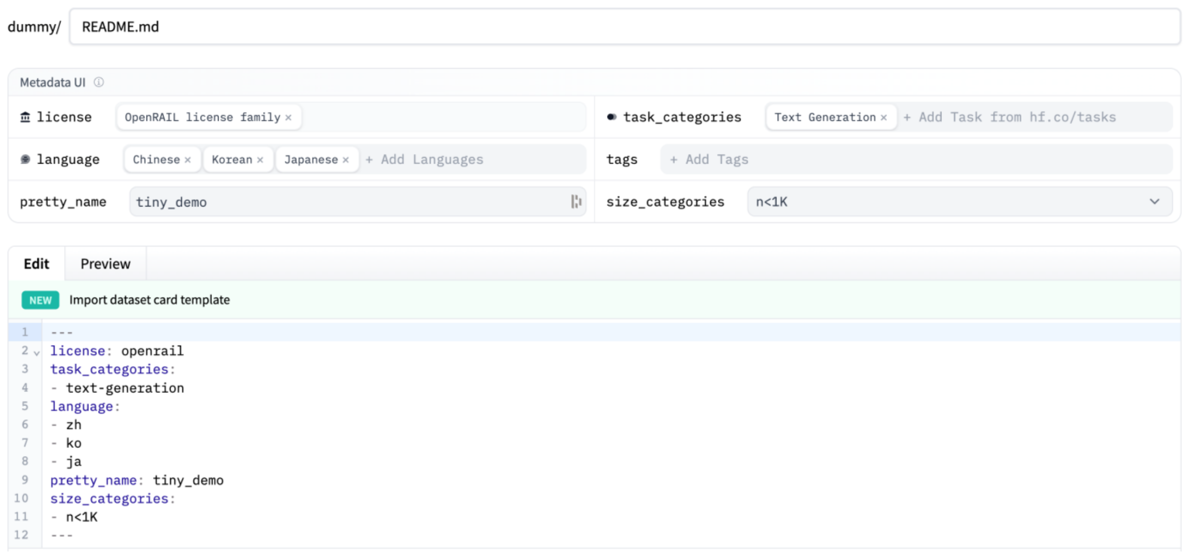
New: Create and edit your dataset card directly on the website!

Models trained or fine-tuned on stevhliu/demo

None yet

2. At the top, you’ll see the **Metadata UI** with several fields to select from such as license, language, and task categories. These are the most important tags for helping users discover your dataset on the Hub (when applicable). When you select an option for a field, it will be automatically added to the top of the dataset card.

You can also look at the [Dataset Card specifications](), which has a complete set of allowed tags, including optional like `annotations_creators`, to help you choose the ones that are useful for your dataset.



dummy/ README.md

license	OpenRAIL license family ×	task_categories	Text Generation × + Add Task from hf.co/tasks
language	Chinese × Korean × Japanese × + Add Languages	tags	+ Add Tags
pretty_name	tiny_demo	size_categories	n<1K

```
---
license: openrail
task_categories:
- text-generation
language:
- zh
- ko
- ja
pretty_name: tiny_demo
size_categories:
- n<1K
---
```

3. Write your dataset documentation in the Dataset Card to introduce your dataset to the community and help users understand what is inside: what are the use cases and limitations, where the data comes from, what are important ethical considerations, and any other relevant details.

You can click on the **Import dataset card template** link at the top of the editor to automatically create a dataset card template. For a detailed example of what a good Dataset card should look like, take a look at the CNN DailyMail Dataset card.

Using the huggingface_hub client library

The rich features set in the `huggingface_hub` library allows you to manage repositories, including creating repos and uploading datasets to the Hub. Visit the client library's documentation to learn more.

Using other libraries

Some libraries like 🤗 Datasets, Pandas, Polars, Dask or DuckDB can upload files to the Hub. See the list of Libraries supported by the Datasets Hub for more information.

Using Git

Since dataset repos are Git repositories, you can use Git to push your data files to the Hub. Follow the guide on Getting Started with Repositories to learn about using the `git` CLI to commit and push your datasets.

File formats

The Hub natively supports multiple file formats:

- CSV (.csv, .tsv)
- JSON Lines, JSON (.jsonl, .json)
- Parquet (.parquet)
- Arrow streaming format (.arrow)
- Text (.txt)
- Images (.png, .jpg, etc.)
- Audio (.wav, .mp3, etc.)
- WebDataset (.tar)

It supports files compressed using ZIP (.zip), GZIP (.gz), ZSTD (.zst), BZ2 (.bz2), LZ4 (.lz4) and LZMA (.xz).

Image and audio files can also have additional metadata files. See the Data files Configuration on image and audio datasets, as well as the collections of example datasets for CSV, TSV and images.

You may want to convert your files to these formats to benefit from all the Hub features. Other formats and structures may not be recognized by the Hub.

Which file format should I use?

For most types of datasets, Parquet is the recommended format due to its efficient compression, rich typing, and since a variety of tools supports this format with optimized read and batched operations. Alternatively, CSV or JSON Lines/JSON can be used for tabular data (prefer JSON Lines for nested data). Although easy to parse compared to Parquet, these formats are not recommended for data larger than several GBs. For image and audio datasets, uploading raw files is the most practical for most use cases since it's easy to access individual files. For large scale image and audio datasets streaming, WebDataset should be preferred over raw image and audio files to avoid the overhead of accessing individual files. Though for more general use cases involving analytics, data filtering or metadata parsing, Parquet is the recommended option for large scale image and audio datasets.

Dataset Viewer

The Dataset Viewer is useful to know how the data actually looks like before you download it. It is enabled by default for all public datasets. It is also available for private datasets owned by a PRO user or an Enterprise Hub organization.

After uploading your dataset, make sure the Dataset Viewer correctly shows your data, or Configure the Dataset Viewer.

3. Materials and Experimental Evaluation

3.1 Dataset

The dataset used for this project is the Cards Image Dataset-Classification, which contains over 8000 images of poker cards in JPG format. The dataset is divided into three sets: training (7624 images), validation (265 images), and testing (265 images). The images are of size 224 x 224 pixels with three color channels.

3.2 Methodology

The project follows a standard machine learning workflow, including the following steps:

1. **Data Collection:** The dataset was downloaded from Kaggle and examined to gain insights into the data.
2. **Data Preprocessing:** The images were preprocessed by resizing them to a standard size and converting them into an array format suitable for feeding into our Mobilenetv2 model.
3. **Data Augmentation:** Data augmentation techniques were used to increase the size of the dataset and improve the robustness of the model. Techniques used include image rotation, flipping, and zooming.
4. **Model Building:** A Mobilenetv2 model was built using Keras with TensorFlow backend, and trained using the preprocessed dataset. Different architectures, hyperparameters, and optimization algorithms were experimented with to achieve optimal performance.
5. **Model Evaluation:** The performance of the model was evaluated using various metrics such as accuracy, precision, recall, and F1 score. The results were visualized using a confusion matrix and classification report.
6. **Model Deployment:** The trained model was deployed to make predictions on new, unseen poker card images. The predictions were visualized.

To run this project, you will need to have the following tools and libraries installed:

- Python 3.8+
- Keras
- TensorFlow
- NumPy
- Matplotlib
- Scikit-learn

You can run the project on your local machine or in a Jupyter notebook environment such as Google Colab.

1. Clone this repository to your local machine:

git clone https://github.com/yourusername/poker-card-image-recognition.git

1. Download the dataset from Kaggle and extract it to the data folder in the project directory.
2. Open the Jupyter notebook poker-card-image-recognition.ipynb and run the cells in order.
3. Once the model is trained, you can use it to make predictions on new poker card images.

Model Architecture

The model architecture consists of a series of convolutional layers followed by fully connected layers. Here's an overview of the model architecture:

Convolutional Layers:

- The model begins with two convolutional layers (conv1 and conv2), which extract features from the input images.
- Each convolutional layer is followed by a Rectified Linear Unit (ReLU) activation function (relu1 and relu2) to introduce non-linearity to the model.
- Max-pooling layers (pool1 and pool2) are applied after each convolutional layer to downsample the feature maps and reduce spatial dimensions.

Fully Connected Layers:

- Following the convolutional layers, the feature maps are flattened and passed through two fully connected layers (fc1 and fc2).
- The first fully connected layer (fc1) has 512 neurons and applies a ReLU activation function (relu3).
- The final fully connected layer (fc2) outputs logits for each class without applying an activation function.

Input and Output:

- Input images are assumed to have three channels (RGB).
- The output layer has num_classes neurons, where num_classes represents the number of classes for classification (default: 53).

Forward Pass:

- During the forward pass, input images (x) undergo convolutional operations, followed by activation functions and max-pooling.
- The resulting feature maps are flattened and passed through fully connected layers to generate class logits.

Overall, the Card Classifier Mobilenetv2 architecture employs convolutional and fully connected layers to learn hierarchical representations of playing card images and make predictions based on these representations.

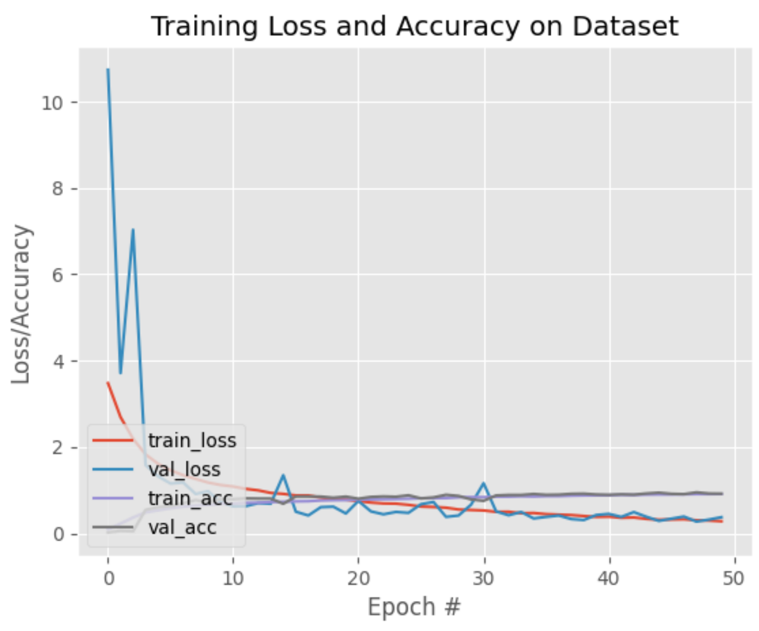
For more details, refer to the Model Architecture section in the code.

Training Process

The model is trained using the Adam optimizer with the Cross-Entropy Loss function. Training is performed over multiple epochs, with early stopping implemented to prevent overfitting. Training progress and performance metrics are monitored using validation data. For detailed information about the training process, refer to the Training Process section in the code.

Model Evaluation

After training, the model is evaluated on a separate test set to assess its performance. The evaluation includes metrics such as accuracy, precision, recall, and F1-score. Additionally, qualitative assessment is performed by visualizing predictions on sample test images.
For more details, refer to the Model Evaluation section in the figure.



Usage

To use the model for inference, follow these steps:

1. Install the required dependencies (specified in the Dependencies section).
2. Clone the repository to your local machine.
3. Download the dataset and place it in the appropriate directory.
4. Run the provided scripts or execute the code in your preferred environment.

Dependencies

Ensure you have the following dependencies installed:

- Python (version 3.9)
- PyTorch (version 2.1.2)
- Matplotlib
- NumPy (version 1.26.3)
- scikit-learn

3.3 Results

Present the quantitative results of your experiments. What are the basic differences revealed in the data. Comparisons to competing methods that address the same problem are particularly useful.(please link your confusion matrix and classification report)

- https://github.com/prameela057/aiclub
- https://aiclub-j5dfwwwok3mkxka2q4ee95.streamlit.app/

3.4 Discussion

What conclusions do the results support about the strengths and weaknesses of your method compared to other methods? How can the results be explained in terms of the underlying properties of the algorithm and/or the data.

4. Future Work

Contributions to this project are welcome. Feel free to open issues, submit pull requests, or provide feedback on the existing implementation.

5. Conclusion

After training all the models for 10 epochs, the performance of **mobilenetv2** surpassed all other models and hence I trained of **mobilenetv2** for 30 epochs to achieve an accuracy value.

6.References

- Kaggle Dataset
- Related Paper

1. I. Vahteristo, Man vs machine: beating humans in a multiplayer card game without lookahead, 2023.

Google Scholar

2. S. Bharany, S. Badotra, S. Sharma, S. Rani, M. Alazab, R.H. Jhaveri, et al., "Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy", *Sustainable Energy Technologies and Assessments*, vol. 53, pp. 102613, 2022.

CrossRef Google Scholar

3.D. Pasquali, J. Gonzalez-Billandon, A.M. Aroyo, G. Sandini, A. Sciutti and F. Rea, "Detecting lies is a child (robot)'s play: Gaze-based lie detection in hri", *International Journal of Social Robotics*, vol. 15, no. 4, pp. 583-598, 2023.

CrossRef Google Scholar

4.B. Sharma and D. Koundal, "Cattle health monitoring system using wireless sensor network: a survey from innovation perspective", *IET Wireless Sensor Systems*, vol. 8, no. 4, pp. 143-151, 2018.

CrossRef Google Scholar

5.L. Fields and J. Licato, "Player Identification for Collectible Card Games with Dynamic Game States", *The International FLAIRS Conference Proceedings*, vol. 36, May 2023.