Predicting Meme Popularity Using Image Classification

Stella Lu
MSCS 2201-2: Artificial Intelligence
Sofia University
Fall 2025

INTRODUCTION

Motivation:
How well do CNNs work on noisy data where social context matters?

Main idea:
Use MobileNetV2 (CNN) to classify reddit memes based on popularity.
We approximate popularity by number of upvotes.

Results:
~45 percent accuracy

- expectedly low
- memes depend on social factors that we did not have data for
- reddit posts are not created equal

DATASET

- ~3.2k samples
- Fields used: media (image URL), ups (upvotes)
- Cleaned results: ~2.4k (some links were dead)

```
Found 3226 meme entries.
Starting image download...


Download completed.
--------------------
Successful downloads: 2387
Failed downloads:     839
```

https://www.kaggle.com/datasets/sayangoswami/reddit-memes-dataset


```
{
    "_default": {
        "1": {
            "title": "Num\ud83c\udd71er One",
            "thumbnail": {
                "thumbnail": "https://b.thumbs.redditmedia.com/FAS_fWvrpmzuPN6Rh67I9ahmovzoe-titgZNilnewpk.jpg",
                "height": 121,
                "width": 140
            },
            "created_utc": 1502621109.0,
            "author": "DrarenThiralas",
            "id": "6tehbc",
            "ups": 87082,
            "downs": 0,
            "media": "https://i.redd.it/7wgs4dkiihfz.png"
        },
        "2": {
            "title": "Got \u2018em",
            "thumbnail": {
                "thumbnail": "https://b.thumbs.redditmedia.com/huIqG6r8L3pryts7WKfLCC-bYHCp2v0dMjXMe2efdPw.jpg",
                "height": 98,
                "width": 140
            },
            "created_utc": 1523557287.0,
            "author": "CasualDad8675309",
            "id": "8bse8k",
            "ups": 75251,
            "downs": 0,
            "media": "https://i.redd.it/65bzzioisir01.jpg"
        },
        "3": {
            "title": "50-0",
            "thumbnail": {
                "thumbnail": "https://b.thumbs.redditmedia.com/1LvJdhkQttbVzoBvGkHaqK-LoQ1yPyKoFr8z-Rc1y2M.jpg",
                "height": 140,
                "width": 140
            },
            "created_utc": 1503862227.0,
            "author": "NikiTosThePleb",
            "id": "6we7gp",
            "ups": 64236,
            "downs": 0,
            "media": "https://i.redd.it/19c4ggoz0ciz.png"
        },
```

BUCKETING

- Converts the task from regression to classification, better for CNNs
- Regression could overfit since we have only ~2.3k images
- Upvotes represented as an integer is more noisy

```
Upvote Distribution:

0-2000: 750
1001-20000: 1222
20001-999999: 1254
```

APPROACH

Pipeline:

1. Load JSON
2. Download images
3. Organize images into /low /medium /high
4. Train a MobileNetV2 classifier
5. Evaluate + visualize results
6. Demo on new memes

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
DATA_DIR = "image_data"

datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,
    validation_split=0.2
)

train_gen = datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="training",
    shuffle=True,
    seed=42
)

val_gen = datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="validation",
    shuffle=False,
    seed=42
)

base_model = MobileNetV2(
    input_shape=IMG_SIZE + (3,),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False
```

```
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation="relu"),
    layers.Dense(train_gen.num_classes, activation="softmax")
])

model.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=10
)

model.save("meme_popularity_model_2.h5")

plt.figure()
plt.plot(history.history["accuracy"], label="train_acc")
plt.plot(history.history["val_accuracy"], label="val_acc")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Training vs validation accuracy")
plt.savefig("accuracy_curve_2.png", dpi=150)
plt.close()

val_gen.reset()
y_true = val_gen.classes
class_indices = val_gen.class_indices
idx_to_class = {v: k for k, v in class_indices.items()}

y_pred_probs = model.predict(val_gen)
y_pred = np.argmax(y_pred_probs, axis=1)

print("Classification report:")
print(classification_report(
```

MODEL ARCHITECTURE

Bullet points or simple diagram:

- MobileNetV2 (frozen)
- Image size: Tried 160 x 160 and 224 × 224
- Added layers:
  - GlobalAveragePooling2D
  - Dense(128, relu)
  - Dense(3, softmax)
- Optimizer: Adam
- Loss: Categorical cross-entropy
- Epochs: Tried both 5 and 10

TRAINING OUTPUT

```
2025-11-27 21:52:38.614117: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 1911 images belonging to 3 classes.
Found 476 images belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_160_no_top.h5
9406464/9406464 [==============================] - 0s 0us/step
Epoch 1/5
60/60 [==============================] - 71s 1s/step - loss: 1.2336 - accuracy: 0.3930 - val_loss: 1.0469 - val_accuracy: 0.4181
Epoch 2/5
60/60 [==============================] - 59s 985ms/step - loss: 0.9914 - accuracy: 0.4966 - val_loss: 1.0340 - val_accuracy: 0.4538
Epoch 3/5
60/60 [==============================] - 59s 987ms/step - loss: 0.9195 - accuracy: 0.5657 - val_loss: 1.0208 - val_accuracy: 0.4370
Epoch 4/5
60/60 [==============================] - 59s 982ms/step - loss: 0.8109 - accuracy: 0.6510 - val_loss: 1.0665 - val_accuracy: 0.4475
Epoch 5/5
60/60 [==============================] - 59s 985ms/step - loss: 0.7045 - accuracy: 0.7122 - val_loss: 1.1454 - val_accuracy: 0.4202
/Users/stellalu/memes/.venv/lib/python3.8/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file fo
rmat is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
15/15 [==============================] - 13s 779ms/step
Classification report:
              precision    recall  f1-score   support

        high       0.55      0.42      0.48       181
         low       0.30      0.47      0.37       122
      medium       0.44      0.39      0.41       173

    accuracy                           0.42       476
   macro avg       0.43      0.42      0.42       476
weighted avg       0.45      0.42      0.43       476
```

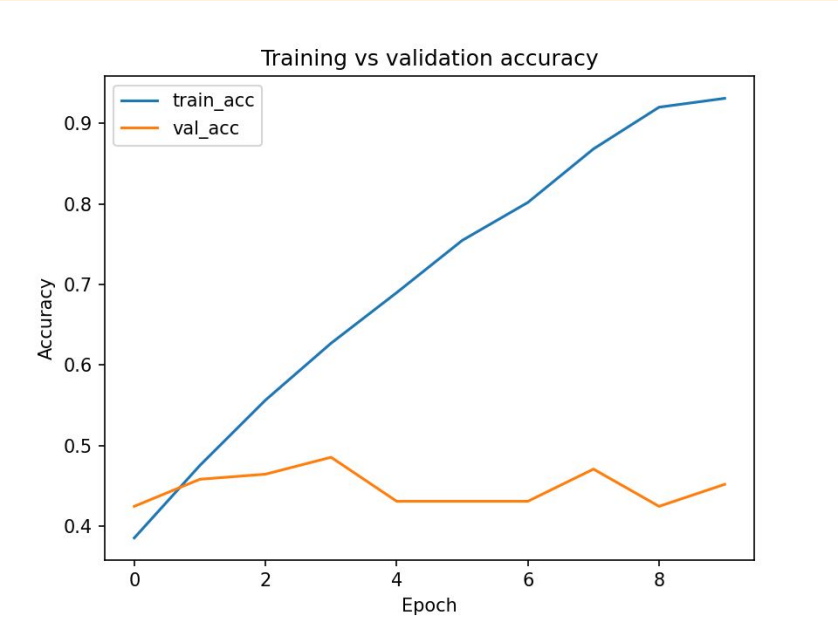
```
2025-11-28 09:48:36.121485: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 1911 images belonging to 3 classes.
Found 476 images belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 [==============================] - 1s 0us/step
Epoch 1/10
60/60 [==============================] - 79s 1s/step - loss: 1.1882 - accuracy: 0.3851 - val_loss: 1.0492 - val_accuracy: 0.4244
Epoch 2/10
60/60 [==============================] - 70s 1s/step - loss: 1.0123 - accuracy: 0.4751 - val_loss: 1.0500 - val_accuracy: 0.4580
Epoch 3/10
60/60 [==============================] - 74s 1s/step - loss: 0.9236 - accuracy: 0.5563 - val_loss: 1.0549 - val_accuracy: 0.4643
Epoch 4/10
60/60 [==============================] - 75s 1s/step - loss: 0.8320 - accuracy: 0.6269 - val_loss: 1.0643 - val_accuracy: 0.4853
Epoch 5/10
60/60 [==============================] - 72s 1s/step - loss: 0.7361 - accuracy: 0.6897 - val_loss: 1.1785 - val_accuracy: 0.4307
Epoch 6/10
60/60 [==============================] - 73s 1s/step - loss: 0.6265 - accuracy: 0.7546 - val_loss: 1.2150 - val_accuracy: 0.4307
Epoch 7/10
60/60 [==============================] - 75s 1s/step - loss: 0.5568 - accuracy: 0.8017 - val_loss: 1.2568 - val_accuracy: 0.4307
Epoch 8/10
60/60 [==============================] - 76s 1s/step - loss: 0.4264 - accuracy: 0.8681 - val_loss: 1.2510 - val_accuracy: 0.4706
Epoch 9/10
60/60 [==============================] - 77s 1s/step - loss: 0.3346 - accuracy: 0.9199 - val_loss: 1.3370 - val_accuracy: 0.4244
Epoch 10/10
60/60 [==============================] - 79s 1s/step - loss: 0.2735 - accuracy: 0.9309 - val_loss: 1.3746 - val_accuracy: 0.4517
/Users/stellalu/memes/.venv/lib/python3.8/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file fo
rmat is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
15/15 [==============================] - 17s 959ms/step
Classification report:
              precision    recall  f1-score   support

        high       0.55      0.57      0.56       181
         low       0.32      0.30      0.31       122
      medium       0.44      0.43      0.44       173

    accuracy                           0.45       476
   macro avg       0.43      0.44      0.43       476
weighted avg       0.45      0.45      0.45       476
```

RESULTS

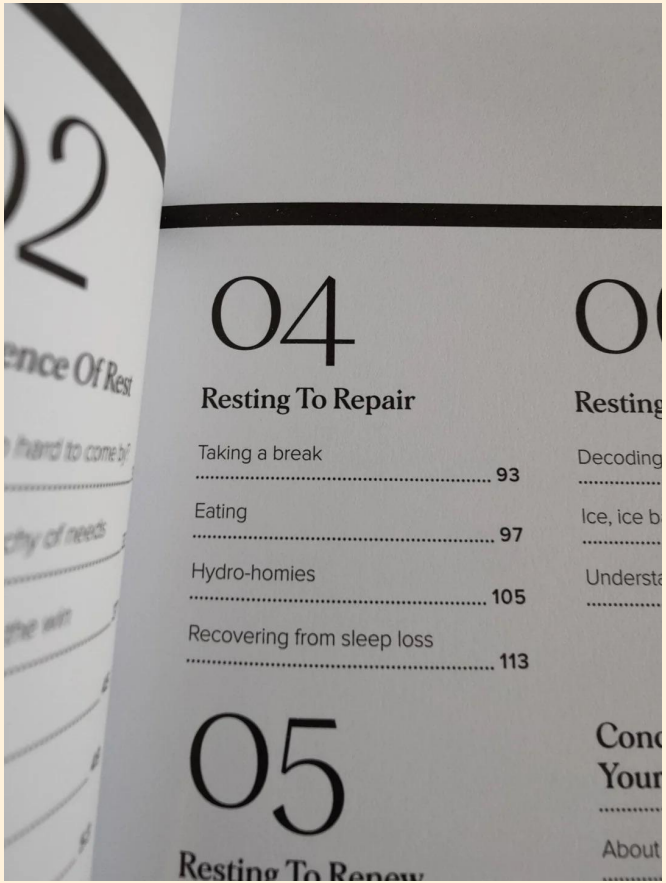
- Training accuracy rose to 93%
- Validation accuracy plateaued ~42% - 46%
- Overfitting: large gap between train and val



DEMO

- 47 (left) vs. 5500 (right) upvotes

source .venv/bin/activate
python demo_predict.py



DEMO

- 47 (left) vs. 5500 (right) upvotes

source .venv/bin/activate
python demo_predict.py

```
Class mapping: {0: 'high', 1: 'low', 2: 'medium'}
1/1 [==============================] - 1s 1s/step

Image: 47_upvotes.png
  P(high): 0.989
  P(low): 0.001
  P(medium): 0.011
--> Predicted label: HIGH
1/1 [==============================] - 0s 52ms/step

Image: 5500_upvotes.jpeg
  P(high): 0.298
  P(low): 0.643
  P(medium): 0.059
--> Predicted label: LOW
```

LIMITATIONS

- Memes include text, cultural references, and timing
- Visuals alone are insufficient
- Dataset is noisy & incomplete
- No text-processing / OCR used

FUTURE WORK

Main goal: Add more context

- Use OCR to incorporate meme text
- Include more metadata (subreddit, date, text/title in the reddit post, reddit tags)
- Fine-tune more layers of MobileNetV2
- Use a vision-language model (CLIP)

QUESTIONS?

Youtube recording: https://youtu.be/PABmCvawu3g

Resources & Interesting Links

- Arxiv: MobileNetV2: Inverted Residuals and Linear Bottlenecks
- Geeks For Geeks: What Is Mobilenet V2?
- Kaggle: Reddit Memes Dataset
- Classifying Popularity Trends of Internet Memes with Machine Learning
- Arxiv: A model for meme popularity growth in social networking systems based on biological principle and human interest dynamics