Selecting Youtube Thumbnails

Hanna Song, Tejas Phirke, Paul Derisemu

Project Questions

- Based on the Youtube Channels, how can we select thumbnails for new videos to be on trending?
- Given the Youtube channels, which thumbnails are considered ‘good’ for getting higher views?
- How CNN could be used on predicting labels of thumbnails?

Introduction

- Predicting if the chosen thumbnail is good or bad to improve the view count and make the video trending.
- Defining Good versus bad: number of views/subscribers
- Mitigate the selection bias:
    - using popular channels over 100M subscribers
    - randomly selecting channels with under 100K subscribers

Modeling Progress

Kaggle and scraping videos from Youtube API

Data collection

Using datasets, train several model with best accuracy

Modeling

Data Preprocessing

Modify the datasets for modeling

Prediction

Using random dataset, predict the image will be good/bad thumbnails

Data Collection

Good thumbnails

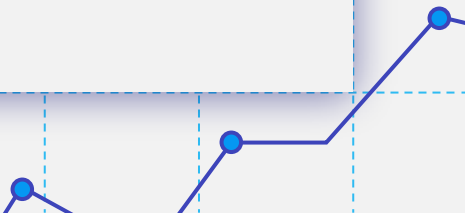
- **Youtube API**
- **Kaggle datasets**
- Popular Channels from different categories

Bad thumbnails

- **Youtube API**
- Randomly selected channels with subscribers under 100K

Random thumbnails

- **Youtube API**
- Randomly selected channels both high and low



Data Preprocessing

Modify datasets

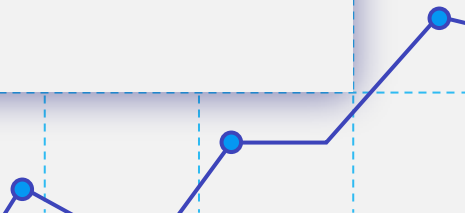
Merging, removing columns and rows

Generate thumbnails

Using video URL, generate thumbnails for both train and test

Label thumbnails

Adding labels for each thumbnails of good and bad



Train the model

Image preprocessing

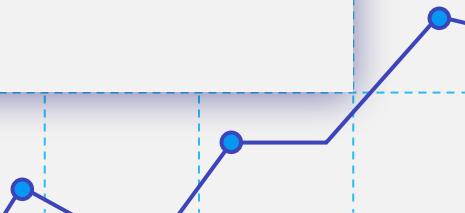
- Normalize images to training form
- Data Augmentation

Subsample of train datasets

Using subsample of training: good thumbnails and bad thumbnails

Test Model

Tested model with accuracy



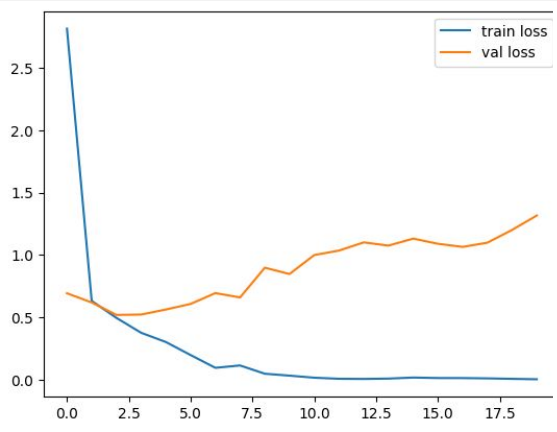
Train the model: Example CNN Code

```python
def preprocess_image(image_path, target_size=(224, 224)):

    image = cv2.imread(image_path)


    image = cv2.resize(image, target_size)


    image = image.astype('float32') / 255.0

    return image


df_train_subsampled = df.sample(n=1000, random_state=42)


df_train_subsampled['Processed_Image'] = df_train_subsampled['Image_Path'].apply(preprocess_image)


label_encoder = LabelEncoder()
df_train_subsampled['Encoded_Label'] = label_encoder.fit_transform(df_train_subsampled['Label'])


X_train_subsampled, X_val_subsampled, y_train_subsampled, y_val_subsampled = train_test_split(
    np.array(df_train_subsampled['Processed_Image'].tolist()),
    df_train_subsampled['Encoded_Label'],
    test_size=0.2,
    random_state=42
)
```

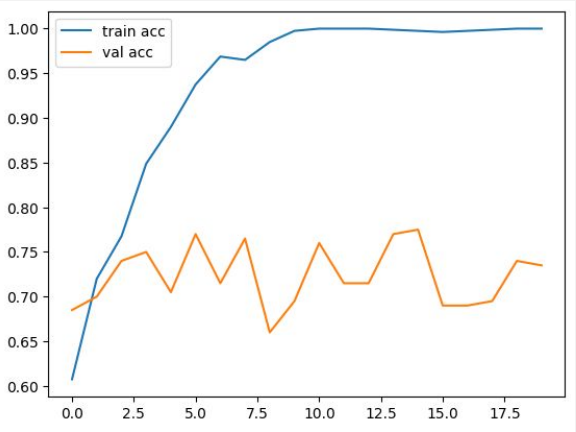
```python
def build_and_compile_model(input_shape=(224, 224, 3), num_classes=1):

    model = models.Sequential()

    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(num_classes, activation='sigmoid'))


    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return model


model = build_and_compile_model()


r =model.fit(X_train_subsampled, y_train_subsampled, epochs=20, batch_size=32, validation_data=(X_val_subsampled, y_val_subsampled))
```

Train the model: Results

Loss



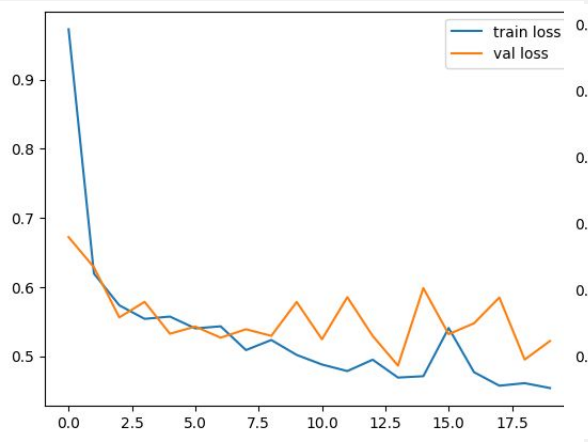
Accuracy



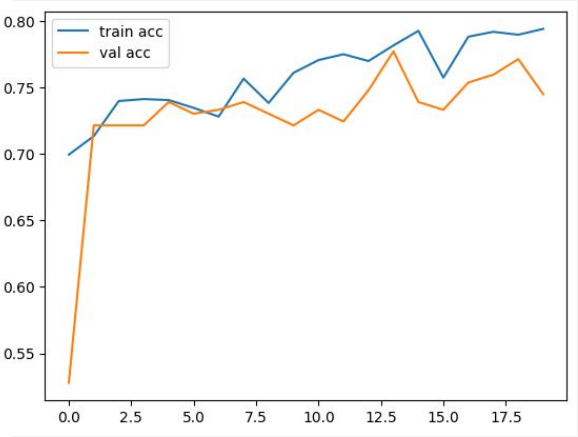
- Loss graph indicates underfitting
- Validation acc not increase while train increase
- ~75%

Train the model: VGG16

Loss



Accuracy



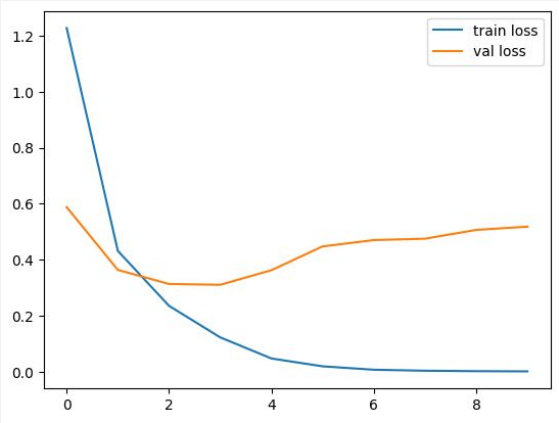
- Using VGG16, both decrease in loss graph – good fit
- Both increase accuracy
- ~73%

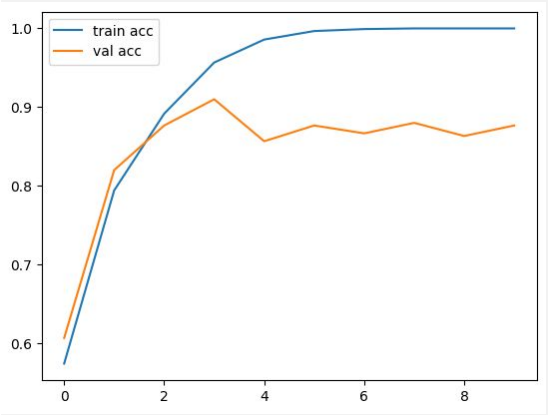
Example:
Prediction on random images to label good or bad thumbnail

Subcategory Modeling

- With our model, we tested on specific category – news to get insight if classifying with video categories can increase the accuracy
- Used frames of the videos for news dataset
- We saw an increase in accuracy by using optimization techniques like learning rate, dropout layer.

- News - Modeling: Results





- Underfitting from loss graph
- Both increase in accuracy for training and validation
- ~93%

Example:
Prediction on random images to label good or bad thumbnail

Image: /content/drive/MyDrive/news/good_news_data_images/video_-sd-780SYAs/frame_0001.jpg
Predicted Label: good

Conclusions

- Our best performance model is VGG16 which indicates good fit and increase accuracy than initial model of CNN
- Dropout gives higher performance on model
- Using categorical labels give a higher prediction
- Using frames rather than thumbnails give a higher prediction

Limitations/Future plans

- Our limitation is on the datasets
  - Unable to get not popular channels as much as we wanted
  - Datasets for testing is also unable to get from various categories
- Future plans
  - Get more datasets
  - Limit the time frame for better labelling
  - With result from news category, we will use frames rather than thumbnails for better accuracy
  - Build an advanced model on VGG16
  - Need increased GPU for VGG16