DESCRIPTION

This is a fun dummy project related to deep learning usage. This is a web to display character name and information in One Piece series. One Piece is one of renowned Anime. It has an interesting story and world development as it has more than 1000 episodes. With that much episodes, we might forget or do not recognize one of them. Thus, this web might be useful for you to know who is character you want to know. However, this web only provides 18 characters to display. Thus, it might also not recognize characters in One Piece.

DATA SOURCE

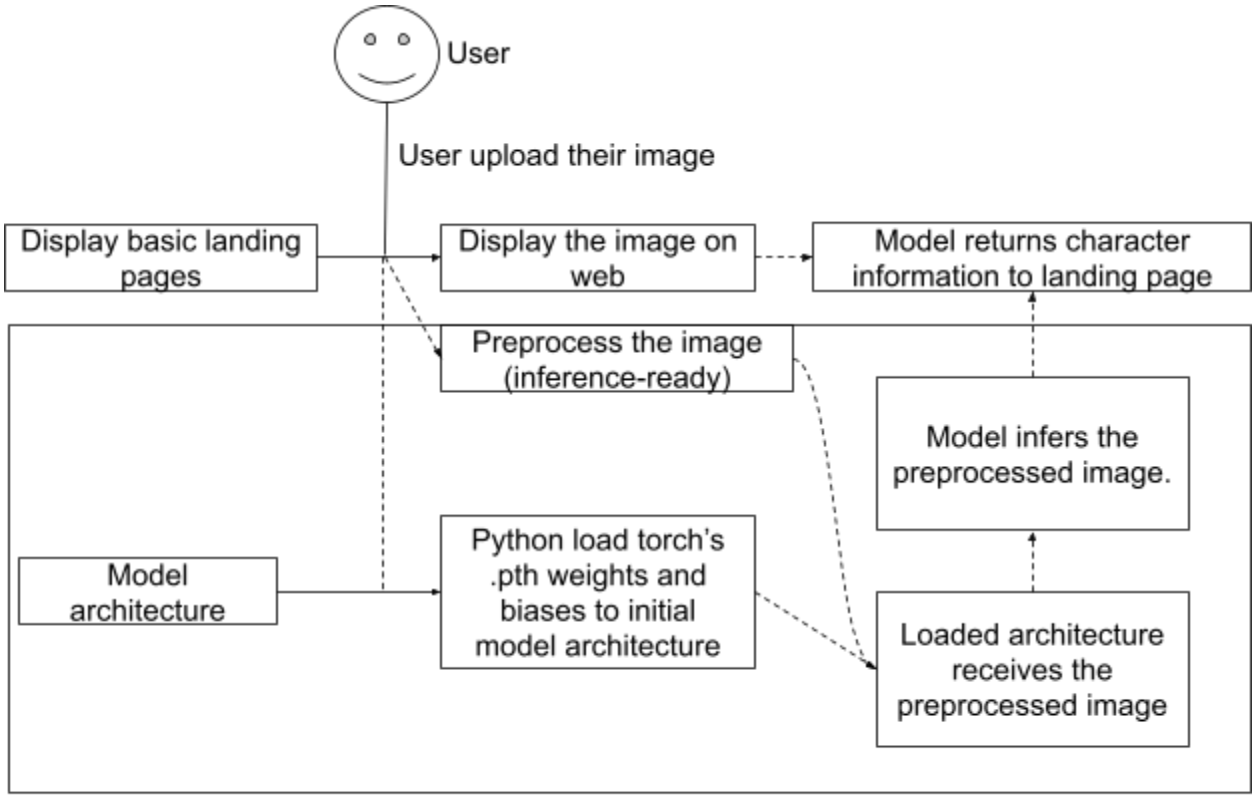
The model is trained based off Kaggle data set, thanks to Ibrahim Serouis. Open this link for more details: https://www.kaggle.com/datasets/ibrahimserouis99/one-piece-image-classifier.

USER FLOW

There are steps to utilize this website:

1. Upload any random One Piece character you found from your favorite search engine
2. Click “Yes, I want!” button to run prediction
3. The model returns its prediction. If it uncertain with its prediction, it will return “Sorry!” messages. If it certain with its prediction, it will return character name and description
4. Upload another image if you want to know another character

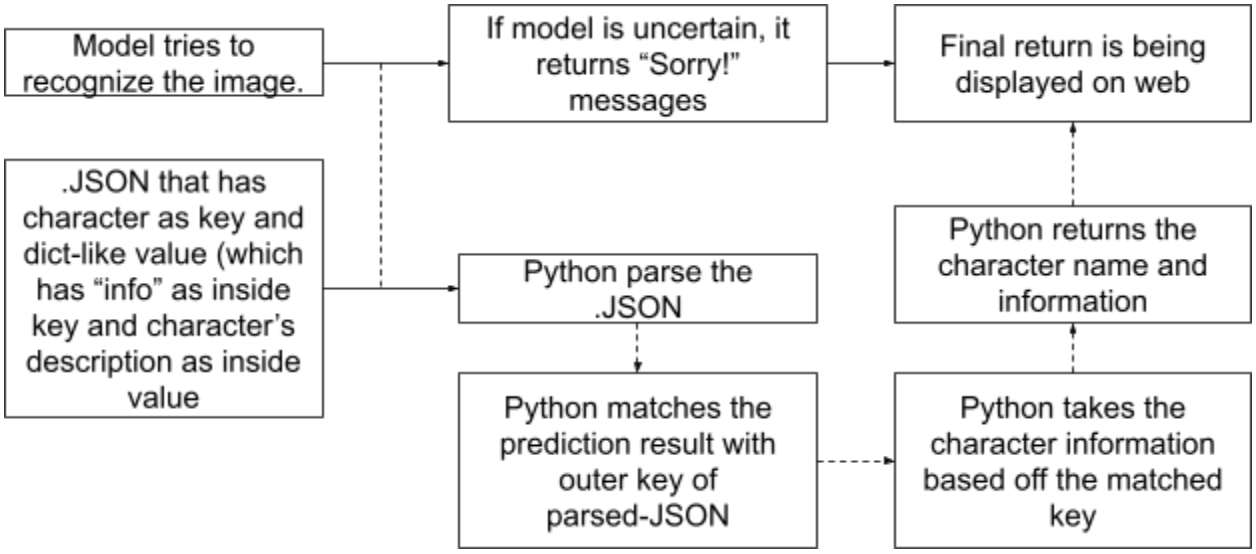
SYSTEM FLOW



Backend flow

Based on image above, after user could have two actions within this web: upload image and click “Yes, I want!” button. First action leads to image being displayed on web and also triggers the appearance of “Yes, I want!” button.

If user follow up their action by clicking that button, it triggers *backend flow* (which is cannot being seen by user). The *backend* flow is being started by preprocessing the image. Preprocessing include resize image pixels and normalize color channels that fit the model architecture. After preprocess the image, Python will load weights and biases in torch’s .pth to fill up the model architecture. If this process is successful (Python is able to identify each layer’s weights and biases), this inference-ready model will receive the preprocessed image. Prediction flow will be illustrated below.



As the model infer (or recognizes) the image, it will return two basic output: character’s name and probability of being that class. For example, it could return “Luffy” with 60% probability. The probability would be extremely useful for several scenarios.

1. If model is uncertain which One Piece character is it (probability: 10% < P(x) < 60%)

   If this scenario happens, a model will return “Sorry!” message but with several follow ups. First, it returns character name that is being predicted. Second, it returns the probability. This is to cross check how uncertain the model is. Since it is uncertain, I will not return any description to not make any misleading understanding.

2. If model is certain which One Piece character is it (probability: P(x) > 60%)

   If this scenario happens, a model will return character name and its description. To display character’s description, Python needs to parse a .JSON file that contains character’s description. After parsing .JSON, Python will find a description that matches the character’s name. While this process is done, Python will map the character name as a second header and its description as a body text that will be displayed on the web. User could use it to recognize further characters! 🙂

3. If model is certain that the image is not One Piece character (probability: P(x) < 10%)

   If this scenario happens, a model will return “Sorry!” message and followed up by a message that tries to notify the user if the image is not a One Piece character.

CAVEATS

1. Since PoV and art styles affect the image’s pixel, our model probably not returning true character. Especially because our model still has ~55% accuracy on test set, which is still bad for generalization.
2. The model is being loaded several times, depends on how much user upload an image and click prediction button. Since it has no “one stop model triggering” (trigger model “activation” only once for continuous actions), the web’s logs may be not “clean” enough.