2018 冬開発合宿

適当に画像収集して
自前の学習データで画像認識

五十嵐 翔

アジェンダ

・モチベーション

・コンテンツ

・デモタイム

・まとめ

↑↑↑↑↑↑↑↑↑↑　発表としてはここまで　↑↑↑↑↑↑↑↑↑↑

・技術ネタ集

モチベーション

・世間の人間がやたらとAIや機械学習など

エンジニアだったら作ってくれるんでしょ？

という風向きをモロに食らってて

最低限の実装と知見を高めたくなってきた。

・個人的作業をもっと自動化していくための仕組みとしても

取り入れたい分野だと感じたから。

コンテンツ

タイトルにも記載されている

学習データをどう作成したのかというと

https://github.com/igara/image_shiki

こちらで画像収集および学習

細かい内容についてはちょっと技術ネタに記載

~~作成した学習データで遊んでみる~~

~~Webアプリとしてこちら~~

~~https://github.com/igara/image-shiki-web-app~~

~~https://syonet.work~~

こちらちょっと作成が間に合わなかったので

夏に持ち越して作成したいなぁと思います。

なので今回の発表は

バッチ形式での画像認識結果をお楽しみください。

とりあえず適当に画像を集めて

4つの分類の学習データを作成してみたので

遊んでみよう

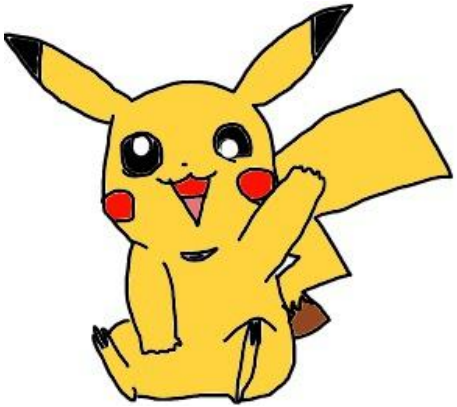
デモタイム

ちょっと手書きトレースした画像や

コラ画像で実際に試してみる

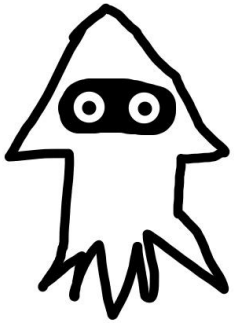
忍者

ピカチュウ

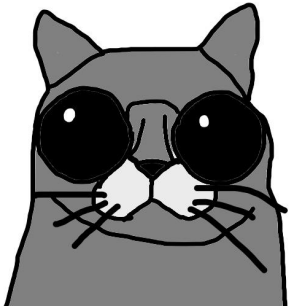


ピカチュウ

ライチュウ



忍者



忍者

あとは人物でやってみたり

まとめ

絵が下手でも色や外見が捕らえられていたら

意外と認識してくれてそう

まるっきり該当しない画像と比較する場合があるので

ノイズとしての画像を集めてやることも大事だと思った

はっぴょう
おしまい

技術ネタ

すみません

メモ書き程度にまとめます。

●画像収集にHeadless Chrome使用して集めてみた

https://github.com/igara/image_shiki

合宿前に準備して作成していたものを使ってみた

方法としてQiitaに書いてたのでこちらをみると良いかも

https://qiita.com/igara/items/e25a5556654e38051559

●画像認識で行なったこと

CNN(Convolution Neural Network　畳み込みニューラルネットワーク)

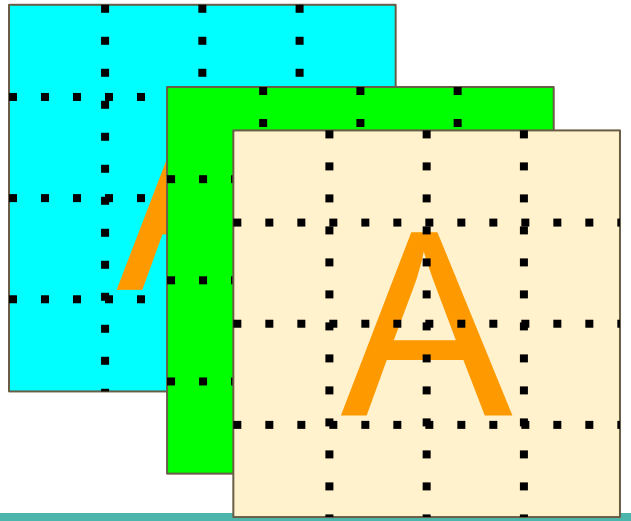
による学習で画像の識別を行なった。

https://github.com/igara/image_shiki/blob/1396197fa28d9657869df1945dba6cfdc38ca65c/save_model.py#L44-L112

すごく要約して言うと集めた画像を無理やり

正方形に縮小し、4 * 4で分割して枚数分

比較して類似性を見ていくようなやり方



●画像認識で行なったこと

Pythonのライブラリであるkerasを使用してみた。

　TensorFlow、Theanoのラッパーライブラリらしい

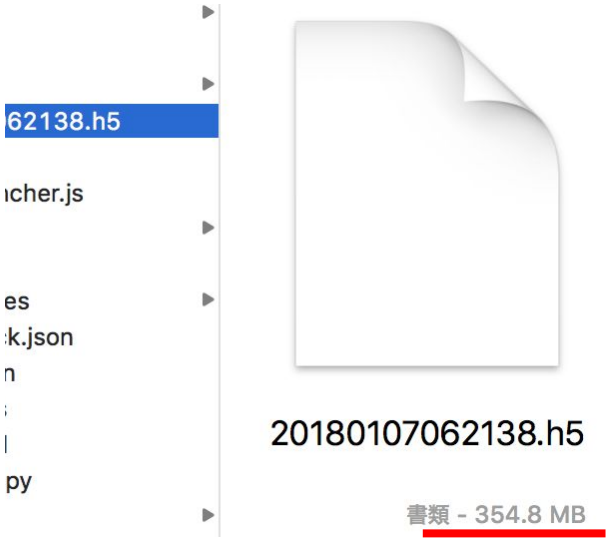
マシンはGPU積んでないmacbookで実施したので

1分類あたりの画像枚数390枚(加工なし)の4分類の

学習済みデータを作成するのに20分弱かかった。

画像全てで約20MB、学習済みデータは350MBも

するのでマシンの性能が欲しくなった。



●なんでdocker使ってるのにvagrantも使用しているのか

某格安VPSのホストOSがUbuntuであったから揃えたかった。

ホスト⇔ゲストのdocker-sync問題

（ファイルシステム的な問題？）

●これ便利と思ったdocker image

https://github.com/SteveLTN/https-portal

local・prod環境でオレオレ・Let's Encrypt 証明書を

ふりわけて作成してくれるだけでなく

nginxも構築してくれるdocker image

設定周りも概ねdocker-compose.ymlに完結できるのでよい

●RustでPython書く不思議なコード　をやろうとしたけど挫折した

https://github.com/dgrunwald/rust-cpython

Rustから認識結果を拾いたかったけど

pipでインストールしたPythonライブラリを使用できなかった。

今回作成したdocker-composeの設定でrust-cpythonを

githubに記載しているサンプルのPythonのバージョン表示するのはできた。

言語バインディング(言語束縛)やりたかった。

●Rustというよりiron　いろいろ挫折した

ironとはRustのWebサーバサイドのライブラリ

iron1つで解決できないことが多い問題に直面する。

ironコミュニティで別で提供されているrouterであったりmountとかを

導入してURLのルーティングまではうまくいっていた。

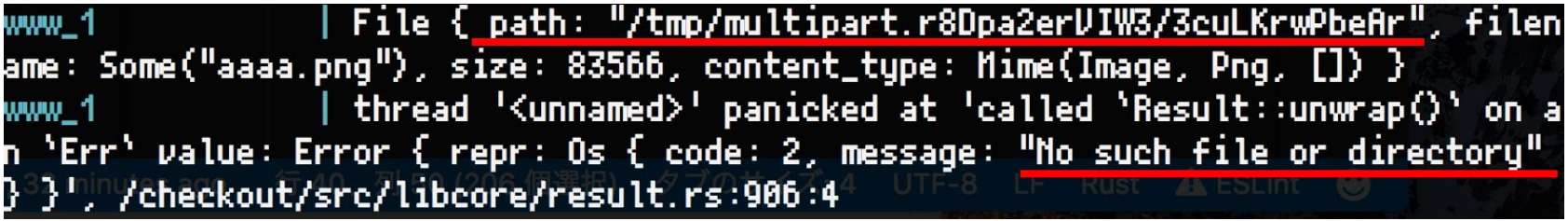
●Rustというよりiron　いろいろ挫折した

リクエストパラメータを取得するためのライブラリparamsを入れてみて

確かにPOSTやGETの時のリクエストデータは取れることを確認できた。

画像のアップロードを試みた時に一時的に保存される/tmp/の画像のパス

返してくれるけど実際には画像データが残っていない問題にあたり積んだ。



●静的ファイルのビルドツールとしてParcel使ってみた

共通部分(common,vender)を作る機構がないようにみえて

ビルドで指定しているエントリーポイント(index.html ...etc)のなかで

vendor、vendorを参照して作る実装部分のファイルと分けてたりすると

静的ファイルの中で案外重複しない感じにモジュール固めてくれる。

●静的ファイルのビルドツールとしてParcel使ってみた

ビルドの出力されたファイルをよしなに呼びやすくなりそうな

謎ビルドツールを作る。

https://github.com/igara/image-shiki-web-app/blob/db8f10e4723052825e17a4874f0960140bc3936a/nodejs/create_parcel_json.js

Parcelで出力されたファイル名がハッシュなものなので

ビルド前のファイル名とハッシュなファイル名をマップにするような

JSONを作成するスクリプトを書いた。

ビルドツール使うならある程度自分でも作る技量持とうって最近思う。

●Mithril.jsはじめてみた

ものすごい雑な感想として

Reactぽくも書けながらStream

http://mithril-ja.js.org/stream.html

による双方向的なバインディングが可能なので

雑に作れそうな感じよかった。

お　し　ま　い