4 Project test and verification

4.1 Natural Language processing model

A screenshot of our data:

	description	Diagnosis
0	male at age of 80.0 . temperature at 100.6 has fever 1.0 with symptoms cough mild or moderate mild or moderate fever temperature 100.6	[FLU, PNEUMONIA, UTI]
1	male at age of 82.0 . temperature at 100.8 has fever 0.0 with symptoms cough mild or moderate fatigue mild or moderate temperature 100.8	[FLU, PNEUMONIA]
2	female at age of 40.0 . temperature at 102.2 has fever 1.0 with symptoms myalgia mild or moderate mild or moderate fever temperature 102.2	[FLU]
3	female at age of 67.0 . temperature at 100.6 has fever 1.0 with symptoms cough mild or moderate myalgia mild or moderate headache mild or moderate mild or moderate fever temperature 100.6	[FLU]
4	male at age of 77.0 . temperature at 101.0 has fever 1.0 with symptoms mild or moderate fever temperature 101	[FLU]
...	...	...
578	male at age of 4.0 . temperature at has fever 4.0 with symptoms cough 4 myalgia 4 headache 4 sudden onset 1 throat 4 4 fever	[FLU]
579	female at age of 7.0 . temperature at has fever 4.0 with symptoms running nose 4 cough 4 sudden onset 1 4 fever	[FLU]
580	male at age of 27.0 . temperature at 98.6 has fever 1.0 with symptoms aches sinus congestion yes fever fatigue temperature 98.6 cough running nose chills vomiting nausea headache high temp home 103 throat	[PNEUMONIA]
581	male at age of 39.0 . temperature at 101.0 has fever 1.0 with symptoms conjunctivitis aches sinus congestion yes fever fatigue temperature 101 cough chills headache high temp home 102 dyspnea	[PNEUMONIA]
582	male at age of 12.0 . temperature at 100.2 has fever with symptoms temperature 100.2 cough throat	[PNEUMONIA]

We will treat this multi-label classification problem as a Binary Relevance problem. Hence, we will now one-hot encode the target variable, i.e., Diagnosis by using sklearn's MultiLabelBinarizer( ).

Now to extract features from the cleaned version of the description data. For this article, I will be using TF-IDF features. I have used the 100,000 most frequent words in the data as my features.

Since we have 28 target diseases, we will have to fit 28 different models with the same set of predictors (TF-IDF features). Hence, I will build a Logistic Regression model **as it is quick to train on limited computational power.**

This is pred_y look like

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

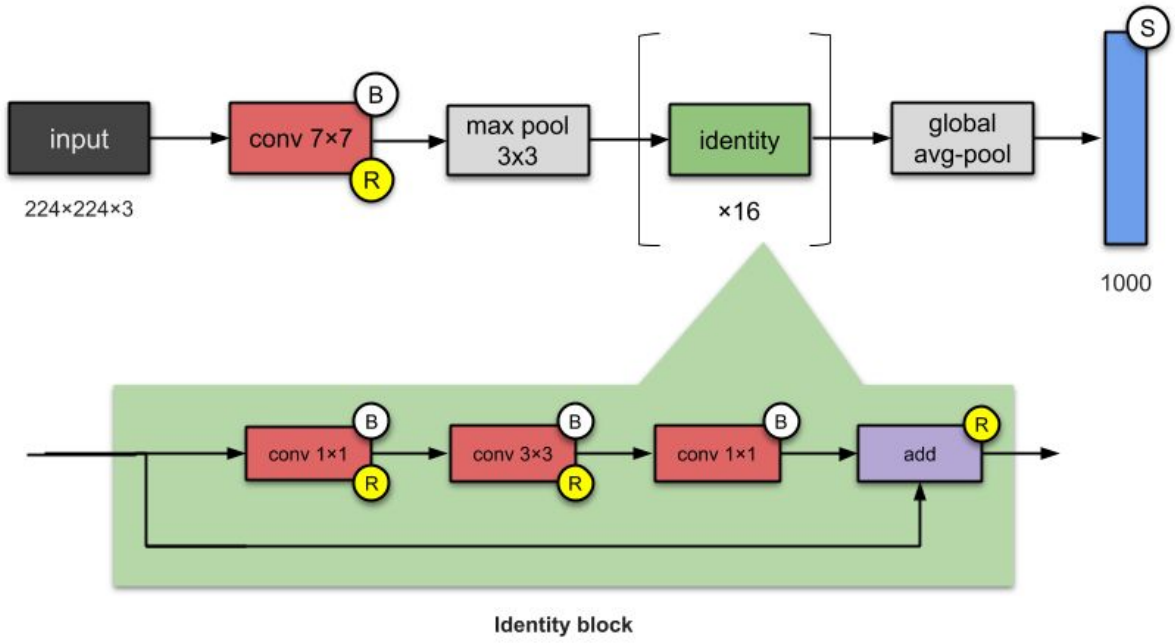
It is a binary one-dimensional array of length 28. Basically, it is the one-hot encoded form of unique diseases.

In order to lower the False Negative Rate(we diagnosed the patient as health when actually she/he has a disease.), we set our threshold as 0.3. After model evaluation and validation, we have 70% accuracy.

4.2 Image detection model

The data using here is the datasets gathered by the organization of ISIC 2019 competition. [1] The data are labeled with one of the nine diagnostic categories: Melanoma, Melanocytic nevus, Basal cell carcinoma, Actinic keratosis, Benign keratosis (solar lentigo / seborrheic keratosis / lichen planus-like keratosis), Dermatofibroma, Vascular lesion, Squamous cell carcinoma, None of the others. We are only the images and the labels of diagnosis of those images in this model building process.

The model we applied is ResNet 50. The structure of the ResNet 50 is shown in the graph, compared to a basic CNN framework:



The modules we used to train this model and apply classification are Pytorch and Fastai. In Train.py, to run this module, we first need to acquire a gpu.

Due to limited computational resources, we haven't finished parameter tuning step in model validation process. The current model we use in the system is the one taking arbitrary parameters and the performance is as following:

Epoch	Train Loss	Validation Loss	Error Rate
0	2.209398	1.848970	0.572797

[1] https://challenge2019.isic-archive.com/

1	1.660940	1.328572	0.474321
2	1.295458	1.231810	0.416052
3	1.089112	1.190999	0.414952
4	0.994005	1.184380	0.402387