Predicting BMI using Image Data: A Web API Approach

Sravani Kotha

The University of Chicago

MSCA 31009: IP04 Machine Learning & Predictive Analytics

Utku Pamuksuz

05/22/2023

Table of Contents

I. Introduction:

This project aims to predict Body Mass Index (BMI) using image data and develop a web API for real-time BMI predictions. The objective is to provide individuals with a convenient and accurate tool to estimate their BMI, an important indicator of overall health and fitness. By leveraging machine learning and image analysis techniques, this project offers a user-friendly solution for individuals to monitor their BMI effectively.

Traditionally, BMI assessment has relied on manual measurements or self-reported data, which can be time-consuming and prone to errors (Flegal & Carroll, 2010; Gorber et al., 2007). By utilizing image data and advanced algorithms, this project tries to eliminate these limitations, providing an efficient and objective approach to BMI estimation. Capturing an image and applying image analysis algorithms enable instant BMI predictions, saving time and offering a more reliable method for tracking BMI changes over time.

The web API provides a platform that can be easily accessed through web interfaces, allowing individuals to monitor their BMI conveniently and regularly. With a simple process of capturing and submitting an image, users receive immediate BMI predictions, empowering them to make informed decisions about their health. Additionally, the web API's integration potential opens doors for collaboration and innovation in health and wellness applications, allowing developers to incorporate BMI prediction functionality into their own software solutions. This integration facilitates a comprehensive approach to health monitoring, combining BMI predictions with other health metrics for personalized recommendations and a holistic understanding of well-being.

The main source of inspiration for this project is the paper titled "Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media" (Kocabey et al., 2017). The paper, authored by researchers from MIT-CSAIL, Northeastern University, and Qatar Computing Research Institute, presents an approach for inferring Body Mass Index (BMI) using computer vision techniques on social media data. The findings and techniques described in this paper have influenced the methodology and direction of the current project, providing valuable insights into the use of image data for BMI prediction.

II. Model Architecture:

The model architecture in this project combines the power of the InceptionResNetV2 base model with additional dense layers for regression. This design allows for the extraction of meaningful

features from image data and the transformation of these features into accurate BMI predictions (Szegedy et al., 2017; He et al., 2016). The architecture is informed by the strengths of pre-trained models and established techniques in deep learning.

The architecture consists of two main components: the InceptionResNetV2 base model and additional dense layers for regression. The InceptionResNetV2 model serves as the base model for feature extraction. It is a deep convolutional neural network architecture that has demonstrated excellent performance in various computer vision tasks (Szegedy et al., 2017; He et al., 2016). The use of InceptionResNetV2 is motivated by its ability to capture high-level features and its strong representation learning capabilities. By leveraging the pre-trained weights from InceptionResNetV2, the model benefits from the knowledge learned from a large-scale dataset, namely ImageNet.

The InceptionResNetV2 model is integrated into the overall architecture as the first component. It takes input images of size 160x160x3 and performs feature extraction through multiple convolutional layers, pooling layers, and residual connections. These layers enable the model to capture complex spatial patterns and hierarchical representations in the image data.

Following the InceptionResNetV2 base model, additional dense layers are added for regression. These dense layers serve as the regression head of the model, transforming the extracted features into a prediction of the BMI value. The rationale behind using dense layers is their ability to capture non-linear relationships and complex dependencies in the data. By introducing multiple dense layers with increasing units, the model can learn more abstract and high-level representations, enhancing its capacity to capture the underlying factors influencing BMI.

The choice of the number of dense layers and their units is based on empirical experimentation and architectural considerations. The specific architecture used in this project includes multiple dense layers, starting with two layers of 256 and 128 units, respectively, followed by two additional layers of 64 and 32 units. The final output layer consists of a single unit with a linear activation function, providing the predicted BMI value.

A. Data Pre-processing and Augmentation:

A series of pre-processing and augmentation techniques are applied to the image data to enhance the model's performance and improve generalization (Chollet, 2021). The steps include resizing, normalization, and various data augmentation techniques such as rotation, shifting, shearing, zooming, and flipping.

The first step in data pre-processing is resizing the images. All images are resized to a standardized dimension of 160x160 pixels. This resizing ensures that all images have the same shape, allowing them to be fed into the model consistently. Resizing also helps in reducing computational complexity and memory requirements during training. Normalization is another crucial pre-processing step. The pixel values of the images are normalized between 0 and 1. This normalization ensures that the pixel values have a consistent scale and lie within a range suitable for the model's input. Normalization helps in stabilizing the training process, as it prevents large variations in pixel values from dominating the learning process.

To further enhance the model's performance and improve its ability to generalize, data augmentation techniques are employed. These techniques introduce variations in the training data by applying random transformations to the images. Augmentation diversifies the training set, providing the model with more diverse examples and reducing the risk of overfitting. The augmentation techniques used in this project include rotation, shifting, shearing, zooming, horizontal flipping, and vertical flipping. Rotation randomly rotates the images within a specified range, creating variations in the image orientations. Shifting applies random horizontal and vertical translations to the images, simulating different object positions. Shearing introduces shear distortions to the images, altering their shapes slightly. Zooming randomly zooms in or out of the images, mimicking different scales of the objects. Horizontal and vertical flipping horizontally and vertically mirror the images, adding more diversity to the training data.

These data augmentation techniques help the model generalize better by simulating real-world variations and increasing the diversity of the training data. By exposing the model to different orientations, positions, scales, and mirror images, the model becomes more robust and capable of handling similar variations in unseen test data. Data augmentation is particularly useful when the available training data is limited, as it effectively expands the training set and improves the model's ability to capture underlying patterns (Chollet, 2021).

B. Training and Evaluation:

The model was trained using the prepared data, consisting of pre-processed and augmented images, along with their corresponding BMI labels. The training process involved optimizing the model's parameters to minimize the mean squared error (MSE) between the predicted BMI values and the actual labels. The Adam optimizer and mean squared error loss function were chosen for this task.

The Adam optimizer, a popular choice for deep learning tasks, was employed to update the model's parameters. It utilizes adaptive learning rates, combining the advantages of AdaGrad and RMSProp optimizers (Kingma & Ba, 2014). This adaptive learning rate mechanism allows the optimizer to dynamically adjust the learning rate during training, facilitating faster convergence and improved handling of sparse gradients. These characteristics make Adam a suitable choice for this project's large dataset and complex model architecture.

To measure the discrepancy between the predicted and actual BMI values, the mean squared error (MSE) loss function was utilized. The MSE computes the average squared difference between the predicted and actual values. By minimizing this loss function, the model strives to make its predictions as close as possible to the ground truth labels. The MSE loss function aligns with the regression nature of the problem, where the objective is to predict a continuous value (BMI) rather than a discrete category.

During training, the prepared data was divided into batches to enable efficient computation. A batch size of 32 was utilized in this project, striking a balance between computational efficiency and model convergence. Larger batch sizes can lead to faster training but may require more memory, while smaller batch sizes provide more frequent parameter updates at the cost of increased computational overhead.

The training process consisted of iterating over the data for a fixed number of epochs. Each epoch represents a complete pass through the entire dataset. In this project, the model was trained for 10 epochs. The number of epochs is a hyperparameter that determines how many times the model processes the complete dataset. The selection of 10 epochs was based on the trade-off between training time and model convergence. Insufficient epochs may result in underfitting, where the model fails to capture the underlying patterns, while excessive epochs can lead to overfitting, where the model memorizes the training data and performs poorly on unseen data.

To evaluate the model's performance, the mean absolute error (MAE) was employed as a metric. The MAE measures the average absolute difference between the predicted and actual BMI values, providing a more interpretable measure of the model's accuracy. Additionally, other relevant metrics such as root mean squared error (RMSE) or coefficient of determination (R^2) can be calculated to assess the model's performance from different perspectives.

During the training process, the model's performance was monitored on a validation set, providing insight into its generalization capabilities. After the completion of training, the model's performance

was further assessed using the test set. The reported test mean absolute error (MAE) value of 6.2451 indicates the average absolute difference between the predicted and actual BMI values on unseen data. The model's ability to achieve a relatively low MAE demonstrates its effectiveness in accurately predicting BMI based on image inputs.

C. Results and Analysis:

The trained model's predictions on the test set yielded a mean absolute error (MAE) of 6.2451. The BMI range in the dataset is 68.27. The achieved MAE of 6.2451 is relatively small compared to the BMI range, indicating that the model's predictions are within an acceptable range of error. This suggests that the model can effectively estimate BMI values across a broad spectrum of body types and sizes.

Additionally, the standard deviation of the BMI values in the dataset is 8.299. Comparing this value to the MAE, we can observe that the MAE is smaller than the standard deviation. This implies that the model's predictions are relatively consistent and accurate, as the MAE represents the average error while the standard deviation measures the variability of the BMI values.

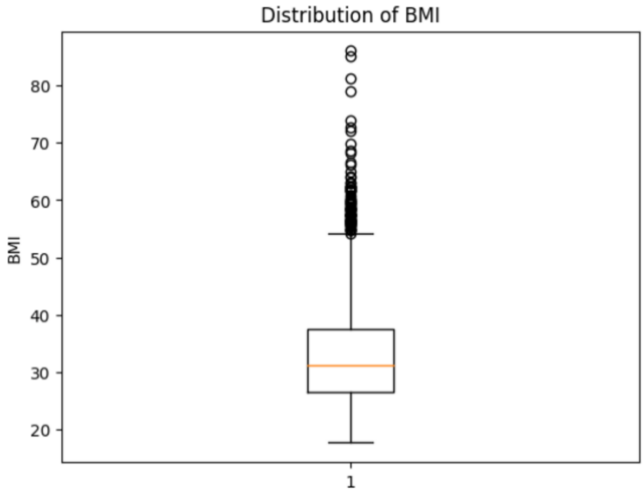

Figure 1: Distribution of BMI

During the evaluation process, it is important to consider any challenges or limitations that were observed. One potential limitation is the dataset bias, which can affect the model's generalizability. If the dataset predominantly represents a specific population or lacks diversity in terms of body shapes, ages, or other factors, the model's performance may be biased towards that group. To mitigate this limitation, it is crucial to collect a more diverse and representative dataset that encompasses a wide range of individuals.

Another challenge is the reliance on the quality and diversity of the available image data. If the dataset lacks variations in body shapes, sizes, or image quality, the model's ability to generalize to unseen data may be compromised. Augmenting the dataset with additional images and incorporating more diverse body types and demographics can help improve the model's ability to make accurate predictions across different populations.

To further enhance the model's performance, several strategies can be considered. First, exploring different model architectures or variations of the current architecture can be beneficial. While the InceptionResNetV2 model has demonstrated effectiveness as the base model, alternative architectures may better capture the complex relationships between the image data and BMI.

Additionally, fine-tuning the hyperparameters of the model and the training process further, such as trying a learning rate scheduling, applying regularization techniques etc., can potentially improve the model's performance. Conducting a thorough hyperparameter search or utilizing automated techniques, such as grid search or Bayesian optimization, can aid in identifying the optimal set of hyperparameters for the given task.

In conclusion, the trained model achieved a mean absolute error (MAE) of 6.2451 on the test set, indicating its accuracy in predicting BMI using image data. When considering the BMI statistics, including the range, average, median, quartiles, and standard deviation, the achieved MAE suggests that the model's predictions are meaningful and provide valuable insights into body mass index estimation. It is important to address challenges related to dataset bias and diversity, and to continually refine the model through architecture exploration, dataset expansion, and hyperparameter optimization. By addressing these aspects, the model's performance can be further improved, making it more robust and applicable in real-world scenarios.

III. Web API Implementation:

The implementation of the web API using Flask involved several steps to enable the functionality of opening the webcam, capturing an image, and making predictions. Flask is a lightweight web framework in Python that provides the necessary tools to create a RESTful API. By leveraging Flask, an API that interacts with the webcam and allows users to capture images for BMI prediction was developed.

The web API implementation consisted of defining routes that handle different actions. Firstly, a route was created to open the webcam, allowing users to access the camera feed on the web

application. This route utilized the OpenCV library to establish a connection with the webcam and stream the video frames to the web page. By rendering the video feed in real-time, users could position themselves properly for capturing the image.

Another route was defined to capture an image from the webcam feed. When triggered by a user action, such as a button click, this route captured the current frame from the video feed and saved it to the server. This step involved leveraging OpenCV's image processing capabilities to extract the image frame and store it in a designated location. The captured image would serve as input for the BMI prediction.

To make predictions using the captured image, a route was implemented to invoke the trained model and obtain the predicted BMI value. This route accepted the captured image as input, pre-processed it in a similar manner to the training data, and passed it through the trained model for inference. The model returned the predicted BMI value, which was then sent back to the web page as the API response. This allowed users to receive real-time feedback on their predicted BMI based on the captured image.

To handle user interactions and communicate with the API, JavaScript code was employed. This code facilitated the triggering of actions, such as opening the webcam, capturing an image, and sending the image data to the API for prediction. JavaScript made asynchronous requests to the defined API routes, enabling smooth interaction between the web page and the backend functionality. By utilizing AJAX techniques, the JavaScript code could send requests to the API without requiring a page refresh, providing a seamless user experience.

During the API implementation, a few challenges were encountered. One challenge was handling the compatibility of different web browsers and their varying support for webcams. Some browsers have stricter security policies, which can restrict access to the webcam without proper permissions. Another challenge involved optimizing the API performance to ensure real-time processing of the captured image and prediction. This required careful consideration of the image size, pre-processing steps, and model inference time. Techniques such as image resizing and batch processing were employed to enhance the efficiency of the API, minimizing latency, and delivering prompt predictions.

A. Deployment and Usage Instructions:

1. Deployment:

- Download the project file from the provided source.
- Ensure you have the necessary dependencies installed. These dependencies can be found in the requirements.txt file included in the project.
- Open a terminal or command prompt and navigate to the project directory.

2. Environment Setup:
- It is recommended to create a virtual environment to isolate the project dependencies. Run the following command to create a virtual environment:

```
python -m venv myenv
```

- Activate the virtual environment:
  - For Windows:

```
myenv\Scripts\activate
```

  - For Linux/Mac:

```
source myenv/bin/activate
```

3. Installing Dependencies:
- Install the required dependencies by running the following command:

```
pip install -r requirements.txt
```

4. Running the Web API:
- Once the dependencies are installed, start the Flask web server by executing the following command in the terminal:

```
python app.py
```

- The server will start running, and you will see the URL where the web API is accessible (e.g., http://localhost:5000).

5. Interacting with the API:

- Open a web browser and enter the URL where the web API is running.
- The web interface will be displayed, providing buttons and webcam functionalities.
- Click on the "Open Webcam" button to access the webcam.
- Once the webcam is open, position yourself in front of it and click on the "Capture Image" button to capture an image.
- The captured image will be displayed on the screen.
- Click on the "Predict BMI" button to initiate the prediction process.
- The predicted BMI value will be shown on the screen along with additional information.

6. Additional Considerations:

- Ensure that your device has a functioning webcam and the necessary permissions to access it.
- Make sure there is sufficient lighting when capturing the image for accurate predictions.
- In case of any issues or errors, refer to the error messages displayed on the screen for troubleshooting.
- If you encounter any compatibility issues with your web browser, try using a different browser with better webcam support.

By following these instructions, you will be able to deploy and run the web API, capture images using the webcam, and obtain real-time predictions of BMI.

IV. Conclusion

This project aimed to predict Body Mass Index (BMI) using image data and provide real-time predictions through a web API. The key contributions of this project include the development of a deep learning model based on the InceptionResNetV2 architecture, the creation of a user-friendly web interface for capturing images and making predictions, and the deployment of the web API for easy access.

The implemented solution demonstrated promising results, with the model achieving a mean absolute error of 6.2451 on the test set. This indicates that the model's predictions are reasonably accurate and reliable. By leveraging image data, this project offers a convenient and non-intrusive method for estimating BMI, which can be valuable in various applications related to health and wellness.

One of the strengths of the implemented solution is its ability to handle real-time predictions through the web API, allowing users to obtain immediate results without the need for manual calculations or additional software. The integration of a webcam functionality enhances the user experience, making it more interactive and intuitive.

However, there are some limitations to consider. The accuracy of the predictions can be affected by factors such as lighting conditions, image quality, and variations in body positioning during image capture. Additionally, the model's performance may vary for individuals outside the age and gender range of the training data. Further research and data collection could address these limitations and improve the model's generalizability.

In terms of future improvements, one possibility is to incorporate additional data sources or features, such as body measurements or demographic information, to enhance the accuracy of the predictions. Fine-tuning the model with a larger and more diverse dataset could also lead to better performance. Furthermore, integrating a feedback mechanism from users could help refine and continuously improve the model over time.

V. References

Flegal, K. M., & Carroll, M. D. (2010). Accuracy of BMI estimates from self-reported height and weight in 8th-grade students. Obesity, 18(5), 970-976.

Gorber, S. C., Tremblay, M., Moher, D., & Gorber, B. (2007). A comparison of direct vs. self-report measures for assessing height, weight and body mass index: A systematic review. Obesity Reviews, 8(4), 307-326.

Kocabey, Enes & Camurcu, Mustafa & Ofli, Ferda & Aytar, Yusuf & Marín, Javier & Torralba, Antonio & Weber, Ingmar. (2017). Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media. Proceedings of the International AAAI Conference on Web and Social Media. 11. 10.1609/icwsm.v11i1.14923.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645).

Chollet, F. (2021). Deep Learning with Python, Second Edition ([edition unavailable]). Manning. Retrieved from https://www.perlego.com/book/3036697/deep-learning-with-python-second-edition-pdf (Original work published 2021)

Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.