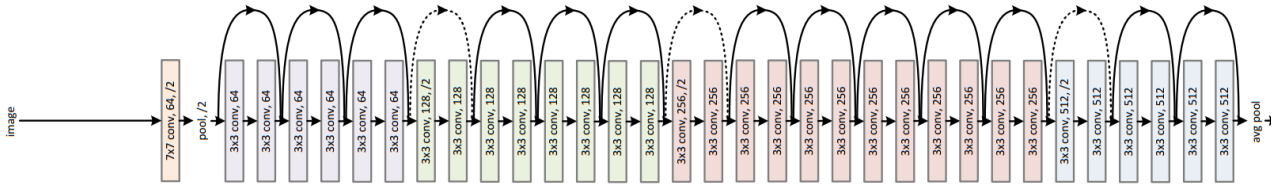
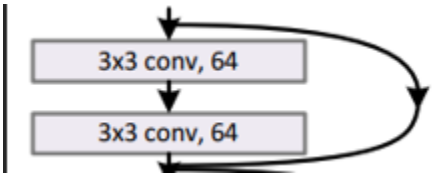
The model applied in this work is basically a ResNet architecture, as introduced in [1512.03385.pdf (arxiv.org)](). The structure of a vanilla ResNet is illustrated as



In the provided code, a BasicBlock refers to a block with residual path like:

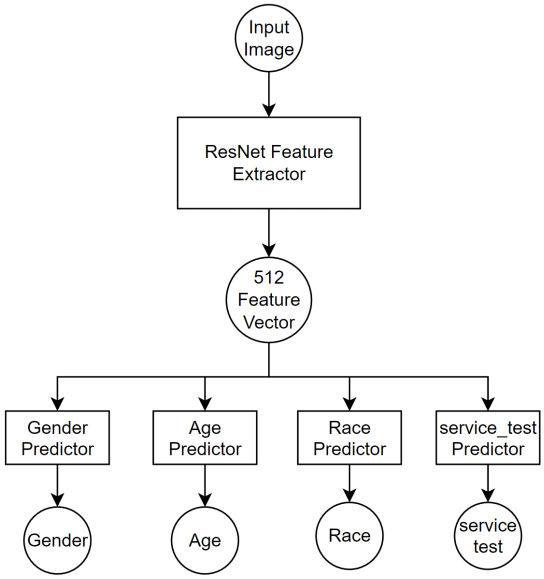


A BasicBlock consists of two convolutional layers with ReLU activation and batch normalization. A BasicBlock doesn't change the size of feature map by setting the kernel size to 3 and padding to 1.

The ResNet class in the code stacks BasicBlock layers to consist the overall ResNet architecture. There are four stacked layers in the ResNet class, where each layer corresponds to a stacked BasicBlock layers, that is, a stacked part in the same color from the graph.

The differences of the ResNet architecture in this work with the vanilla ResNet is that to reduce the time consumption, the only 2 residual blocks are used per stack, that is, the structure becomes.

Then, given that this is a multi-task prediction task to predict gender, age, race and the service_test column, all the predictor layers are built upon the ResNet feature extractor to make a more powerful feature extractor which leverages all the label information. Thus, four TaskHead are applied upon the ResNet output as illustrated below:



Each head is a 2-layer MLP network and is a multi-class classification network. That is, there are four multi-class classification tasks in total and thus four cross-entropy loss functions are applied on each task. The model is optimized to minimize the sum of the four loss functions with an Adam optimizer.

Each predictor in the graph is implemented by the class TaskHeads in the code file. CustomModel class composes the ResNet feature extractor and all the four predictors as the overall structure.

The CustomDataset class provide the entry for the model to read data from. Given an index, a CustomDataset read the corresponding image file from the train or val folder and retrieve labels from the fairface_label csv file.

To make the model able to process the labels as numerical inputs, all the labels are converted into number 0, 1, 2, and so on. The mapping for each task is built in variables like age_to_num and num_to_age.

The model is trained for 20 epochs in a for loop:

```
for epoch in range(num_epochs):
```

In each epoch, the model is trained on the training set and validated on the validation set. For the validation set, the number of correct predictions of each task are recorded to calculate the accuracy.

The trained model is saved in a `multi-task.ckpt` file so it can be loaded without training for further evaluation steps.