Project Proposal - Ready to Use Models in MLpack

Name: Aakash Kaushik

University: SRM Institute of Science and Technology, Chennai, India

Field of study: B.Tech in Computer Science Engineering

Date study was started: June 2019

Expected graduation date: May 2023

Homepage: https://aakash-kaushik.github.io/Aakash-kaushik/

Email: kaushikaakash7539@gmail.com

IRC nick: Aakash kaushik @Aakash-kaushik (Gitter)

Interests and hobbies:

1. Deep Learning - I learned about Deep Learning when I came to university. It intrigued me and since then I've had a few internship opportunities as a Deep Learning researcher and engineer.
2. Music - I tend to be more into pop, but music is something I listen to every day, but yeah, I wasn't listening to it while I was writing the application.
3. Psychology - Just a general interest in psychology, fascinated by the things that influence how we make conscious and unconscious decisions.

Q: What languages do you know? Rate your experience level (1-5: rookie-guru) for each.

The languages I know are Python and C++. I would rate my experience level to be 3.5 and 4 respectively.

Q: How long have you been coding in those languages?

I have been coding in C++ for 3.5 years and in python for about 1.5 years.

Q: Are you a contributor to other open-source projects?

Yes, but I haven't contributed to those projects as much as compared to mlpack.

Q: Do you have a link to any of your work (i.e. Github profile)?

Link to Github profile.

Q: What areas of machine learning are you familiar with?
I am familiar with basic Machine learning and Deep learning in which I am proficient with computer vision and familiar with NLP.

Q: Have you taken any coursework relevant to machine learning?
Yes, I took the following coursework.

1. Machine Learning, Stanford University on Coursera
2. Deep Learning Specialization, deeplearning.ai on Coursera
3. Machine learning with Google Cloud Platform on Qwiklabs

Q: What are your long-term plans, if you have figured those out yet? Where do you hope to see yourself in 10 years?
As for my long-term plans, I would like to work as a deep learning engineer, work with models, put them into practice, solve the problems that arise in training and deployment, and help develop deep learning libraries. I've thought about it, but that might change as I grow and learn more about the field.

Q: Describe the most interesting application of machine learning you can think of, and then describe how you might implement it.
The idea was to implement a vision system to control things in my room, it would consist of two things:

1. Face recognition - So that it doesn't register everyone's gestures.
2. Hand pose estimation model - To recognize the gestures.

These two parts solve the identity and gesture recognition problem. With the API for Google Assistant, I could then just pass the recognized gestures and control some things in my room.
Since the Assistant API was the only component I wanted to run online, both models had to be lightweight because of this limitation since I wanted to run them on a Raspberry Pi, but at the time I didn't have that much money and so this idea never really came to life.

Q: Both algorithm implementation and API design are important parts of mlpack. Which is more difficult? Which is more important? Why?
I think implementing the algorithm is harder from a developer's point of view, but when it comes to an end-user, both the algorithm and the API are components that are important because if you have a non-intuitive API that the end-user can't cope with, they just won't use it, They just keep popping up and the algorithm is important to the end user because assuming they can find their way around your API but don't get the results they expect or the algorithm is too slow or too hard for their machine, as is often the case with deep learning algorithms, they just won't be able to use it.

Project Proposal

The Objective in brief ...

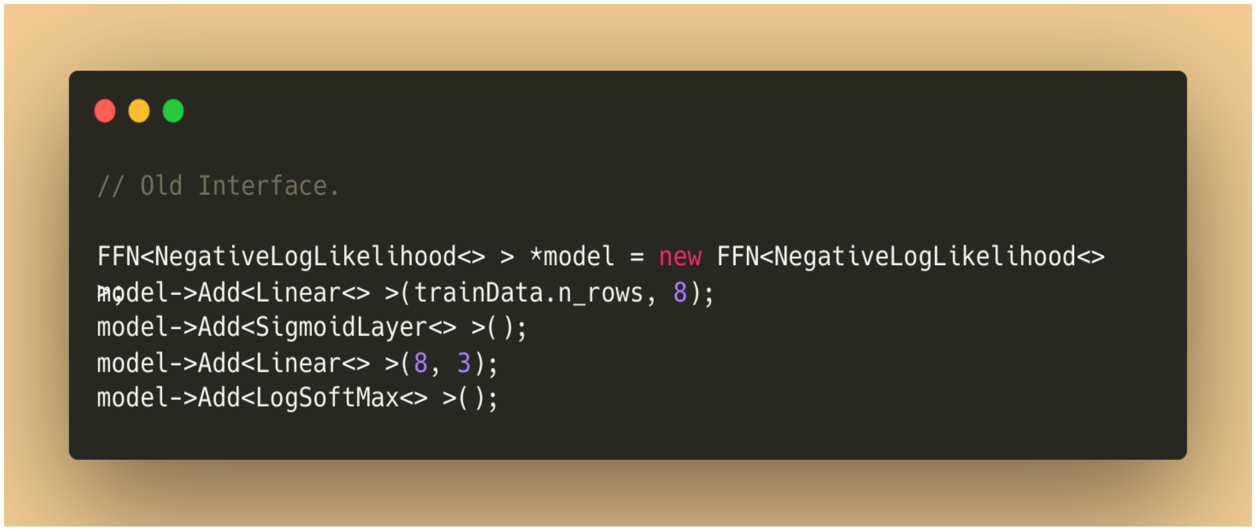
Is to implement **MobileNetV1** which will also include implementing **depthwise separable convolutions** and a resnet model builder that can be used to create **resnet18**, **resnet34**, **resnet50**, **resnet101**, and **resnet152**. This project would fall under the idea: **Ready to use models in mlpack**, the resnet builder and MobileNetV1 will fall under the models repository and depthwise separable convolutions will fall under the mlpack repository as a layer inside the artificial neural network codebase.

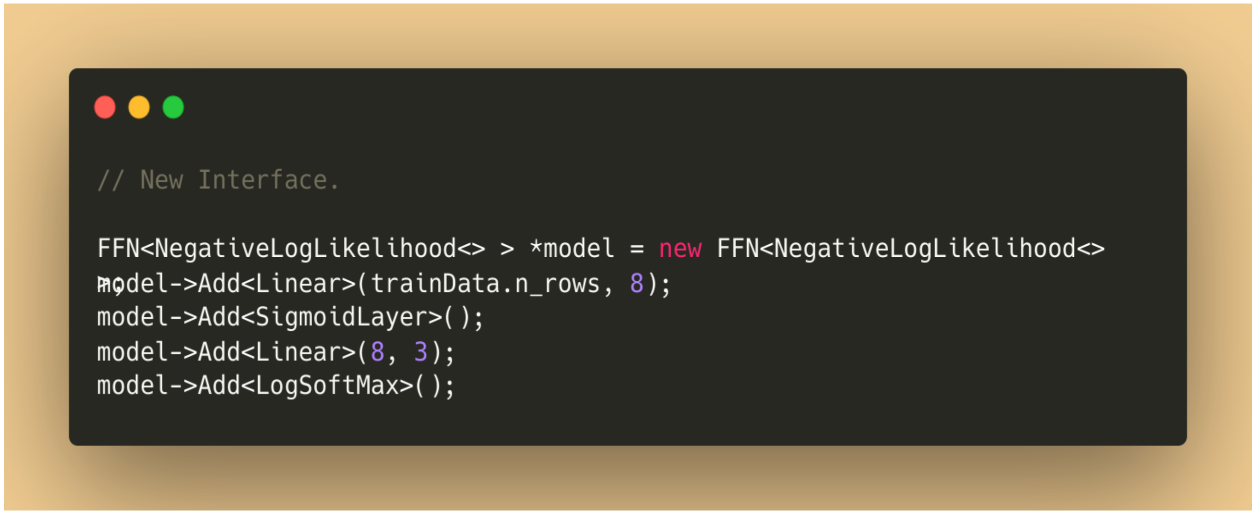
Contributions to mlpack as an organization

In this section, I would like to go over some of the contributions I have had and they have and would help me in my project for this summer.

- **Swap boost::variant with vtable**

The most notable contribution that will help me in this project is the one where we are replacing boost::variant with vtable and I was fortunate enough to do the groundwork for that which introduced me to all the layers that exist in the ANN codebase in mlpack and also how they are stored, called and manipulated.
So, for the user-facing side there is not much change. I have demonstrated an old vs new interface for the user.

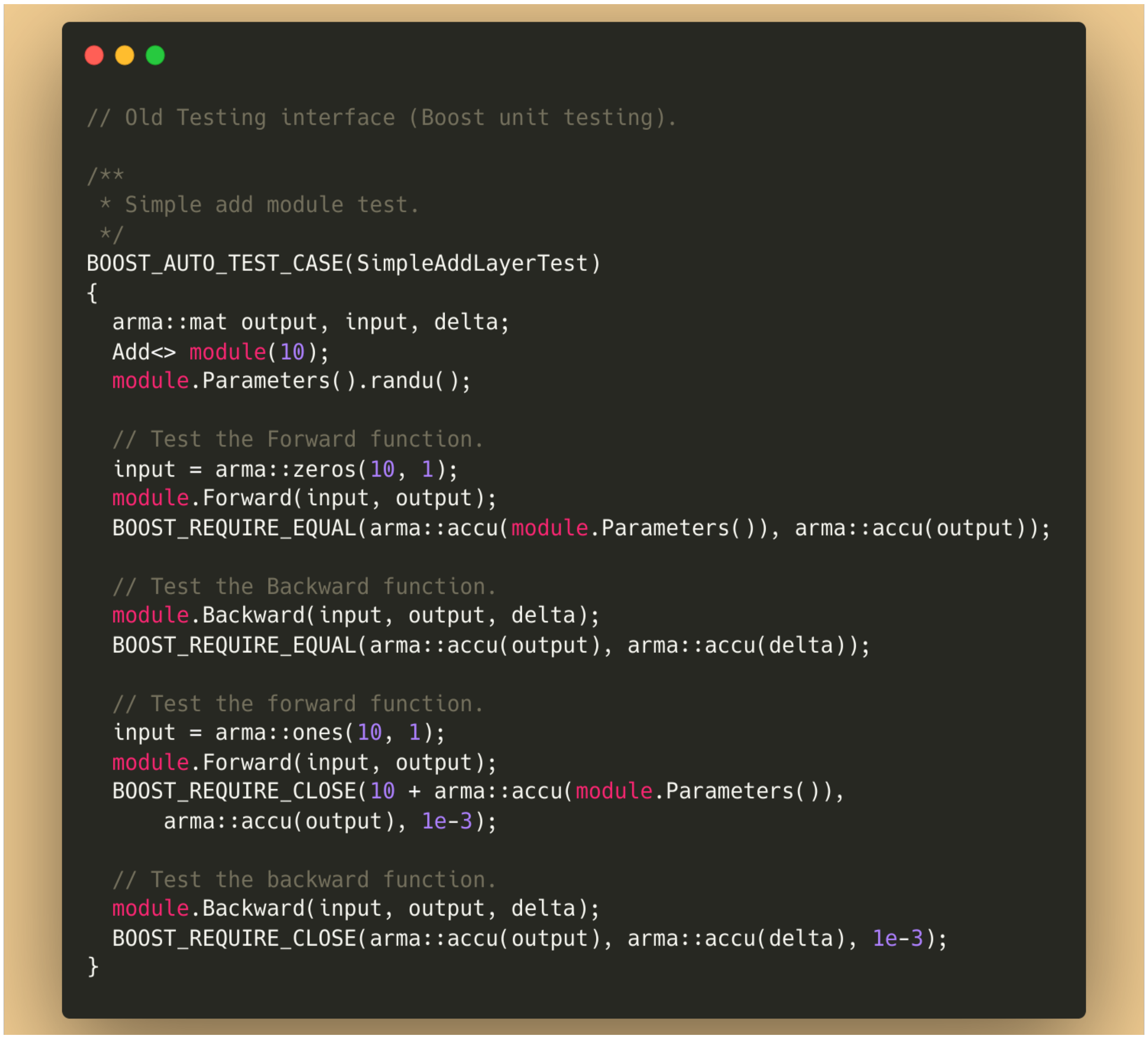

```
// Old Interface.

FFN<NegativeLogLikelihood<> > *model = new FFN<NegativeLogLikelihood<>
model->Add<Linear<> >(trainData.n_rows, 8);
model->Add<SigmoidLayer<> >();
model->Add<Linear<> >(8, 3);
model->Add<LogSoftMax<> >();
```


```
// New Interface.

FFN<NegativeLogLikelihood<> > *model = new FFN<NegativeLogLikelihood<>
model->Add<Linear>(trainData.n_rows, 8);
model->Add<SigmoidLayer>();
model->Add<Linear>(8, 3);
model->Add<LogSoftMax>();
```

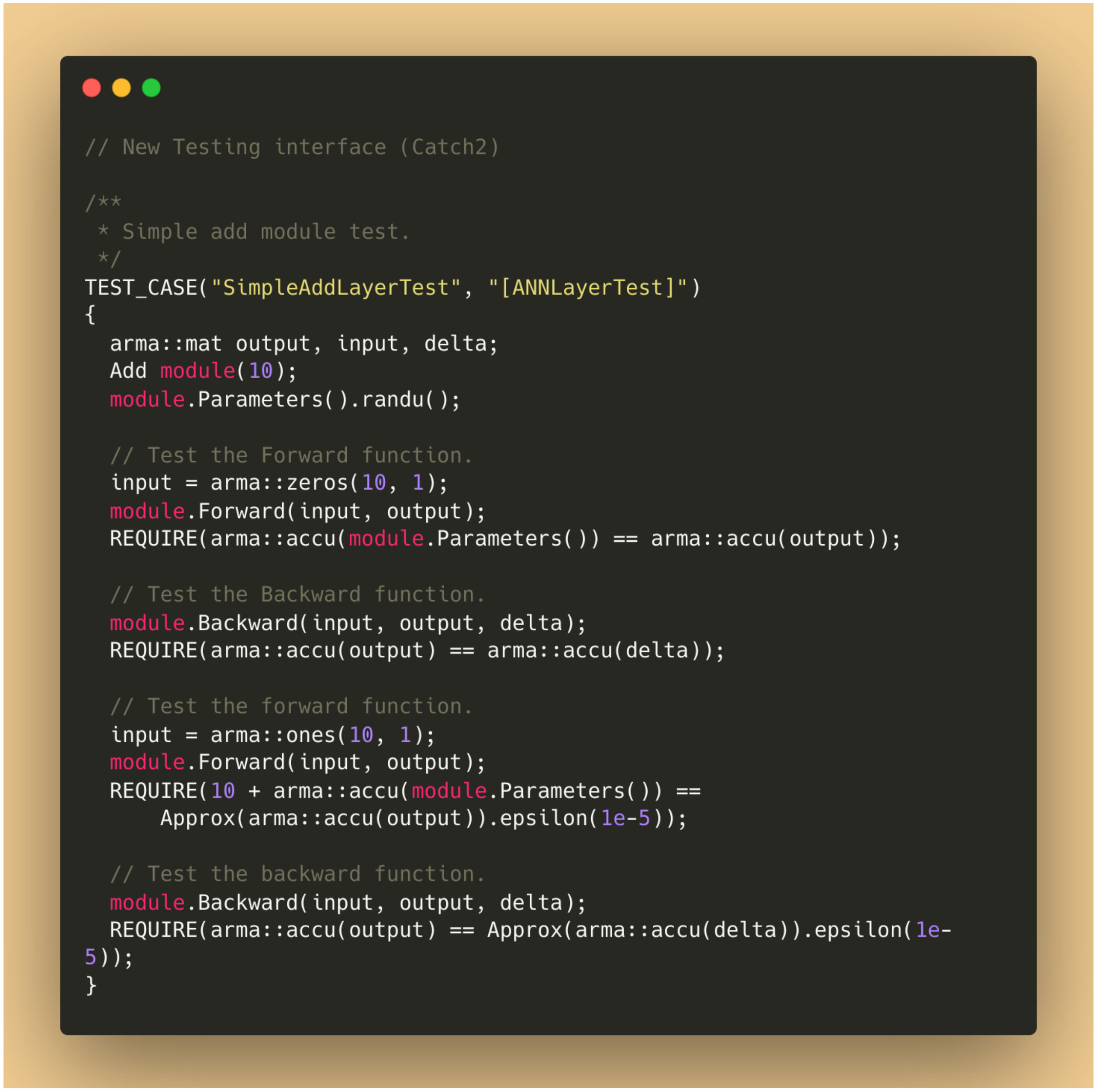
And that’s it for the user you might have even missed the change if you were reading too quickly it is just that we now provide a typedef for all the layers so you don’t need to call them with the empty angular brackets for template params, this is just an addition to the PR for swapping boost variant with vtable the actual work was done inside where now we have a base class and we use inheritance to call all the layers through the base class pointer.
My contributions to this were that I did the groundwork to port the FFN class and the linear layer which is mentioned in [#2777](#2777) and on further I proceeded to make another PR directly to zoq/mlpack/ann-vtable which ports all the remaining classes which are RNN and BRNN and all the layers that were remaining and modified tests to work with the new API all of which can be seen here in [#1](#1).

- **Porting tests from boost to Catch2**

There had been numerous changes to reduce the compile time footprint of mlpack, the above-explained PR was also a part of that, and removing boost unit testing was also a part of that in place of that we now use Catch2 for testing. The major part of this was to change the syntax used for all the tests that exist in mlpack because boost unit testing and Catch2 didn't use the same syntax. An example of the type of change is demonstrated below.



```cpp
// Old Testing interface (Boost unit testing).

/**
 * Simple add module test.
 */
BOOST_AUTO_TEST_CASE(SimpleAddLayerTest)
{
  arma::mat output, input, delta;
  Add<> module(10);
  module.Parameters().randu();

  // Test the Forward function.
  input = arma::zeros(10, 1);
  module.Forward(input, output);
  BOOST_REQUIRE_EQUAL(arma::accu(module.Parameters()), arma::accu(output));

  // Test the Backward function.
  module.Backward(input, output, delta);
  BOOST_REQUIRE_EQUAL(arma::accu(output), arma::accu(delta));

  // Test the forward function.
  input = arma::ones(10, 1);
  module.Forward(input, output);
  BOOST_REQUIRE_CLOSE(10 + arma::accu(module.Parameters()),
      arma::accu(output), 1e-3);

  // Test the backward function.
  module.Backward(input, output, delta);
  BOOST_REQUIRE_CLOSE(arma::accu(output), arma::accu(delta), 1e-3);
}
```



```
// New Testing interface (Catch2)

/**
 * Simple add module test.
 */
TEST_CASE("SimpleAddLayerTest", "[ANNLayerTest]")
{
  arma::mat output, input, delta;
  Add module(10);
  module.Parameters().randu();

  // Test the Forward function.
  input = arma::zeros(10, 1);
  module.Forward(input, output);
  REQUIRE(arma::accu(module.Parameters()) == arma::accu(output));

  // Test the Backward function.
  module.Backward(input, output, delta);
  REQUIRE(arma::accu(output) == arma::accu(delta));

  // Test the forward function.
  input = arma::ones(10, 1);
  module.Forward(input, output);
  REQUIRE(10 + arma::accu(module.Parameters()) ==
      Approx(arma::accu(output)).epsilon(1e-5));

  // Test the backward function.
  module.Backward(input, output, delta);
  REQUIRE(arma::accu(output) == Approx(arma::accu(delta)).epsilon(1e-
5));
}
```

The contributions I did for this was to port a major number of tests from boost unit testing to Catch2, which can be seen in these PRs: #2678, #2644, #2640, #2638, and #2627.

- **Porting tests from boost to Catch2 in models repository**

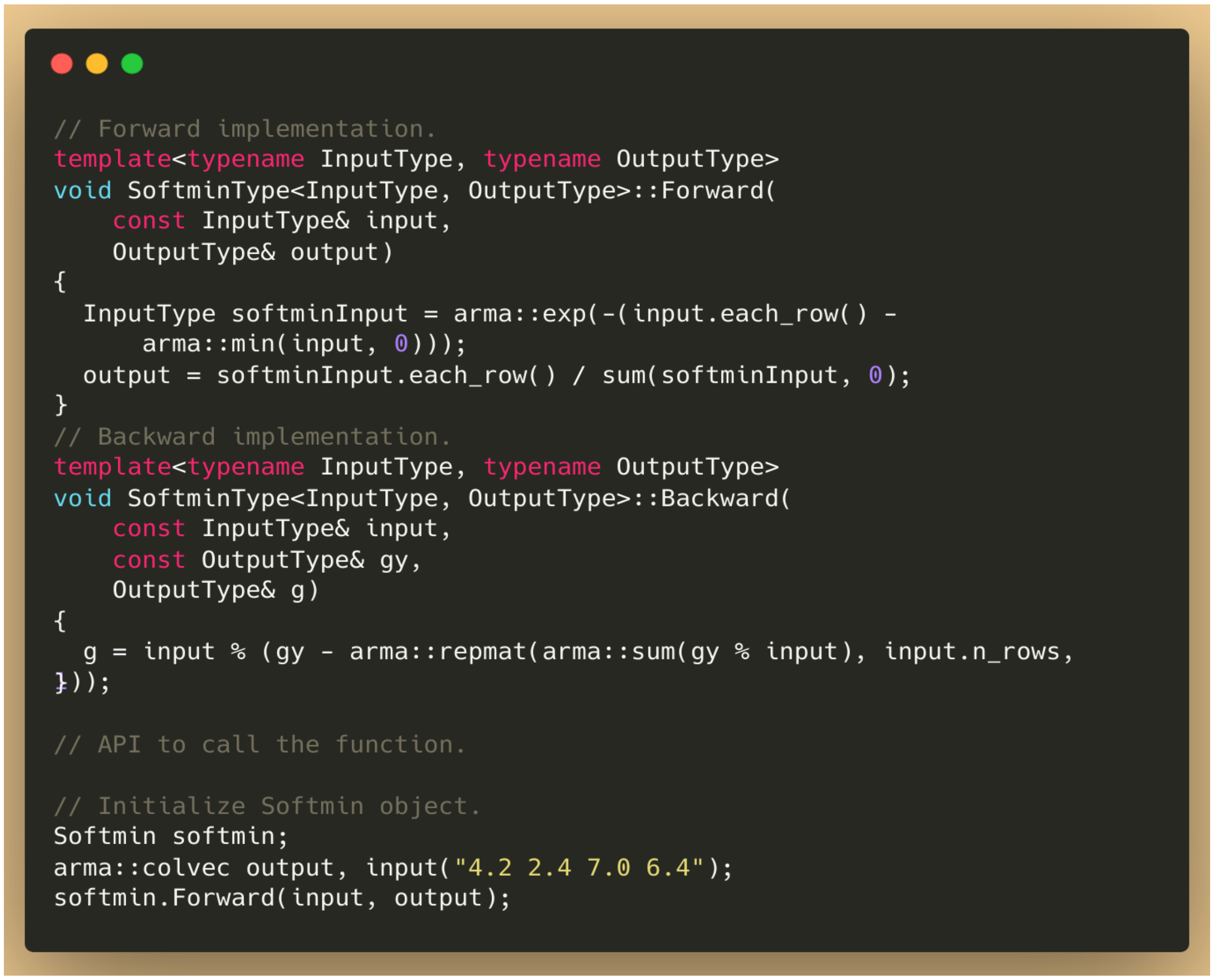
As mlpack had many changes, the model repository wasn't able to "catch" up to it and was still using boost unit testing so it was the same kind of work as explained above.

My contributions in this were that I had to set up the Catch2 testing suit from the start and port all the tests in the models repository and make according to changes which can be seen here: #52.

- **Softmin Activation function**

Mlpack lacked the implementation of the softmin activation function for which the formula can be seen below and can be read more about here.

$$f_i(x) = \frac{\exp(-x_i)}{\sum_j \exp(-x_j)}$$



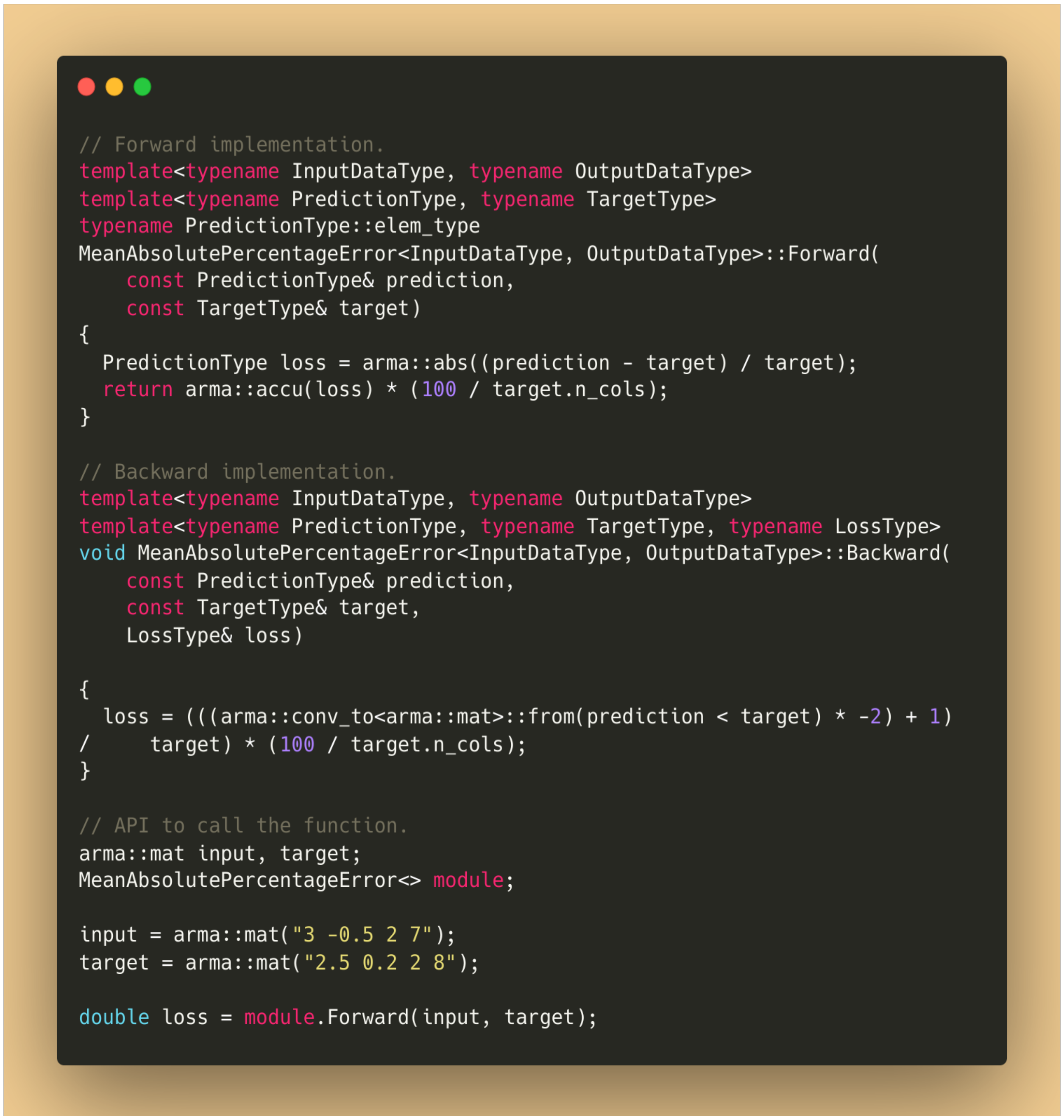
```
// Forward implementation.
template<typename InputType, typename OutputType>
void SoftminType<InputType, OutputType>::Forward(
    const InputType& input,
    OutputType& output)
{
  InputType softminInput = arma::exp(-(input.each_row() -
      arma::min(input, 0)));
  output = softminInput.each_row() / sum(softminInput, 0);
}
// Backward implementation.
template<typename InputType, typename OutputType>
void SoftminType<InputType, OutputType>::Backward(
    const InputType& input,
    const OutputType& gy,
    OutputType& g)
{
  g = input % (gy - arma::repmat(arma::sum(gy % input), input.n_rows,
}));

// API to call the function.

// Initialize Softmin object.
Softmin softmin;
arma::colvec output, input("4.2 2.4 7.0 6.4");
softmin.Forward(input, output);
```

My contributions here were to include the softmin activation function, verify the output with Pytroch and write tests to verify the function's integrity and correct implementation. This can be seen in PR: #2621.

- **Mean Absolute Percentage Error**

Mlpack also lacked MAPE as a loss function which the formula can be seen below and can be further read about here on Wikipedia.

$$M = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

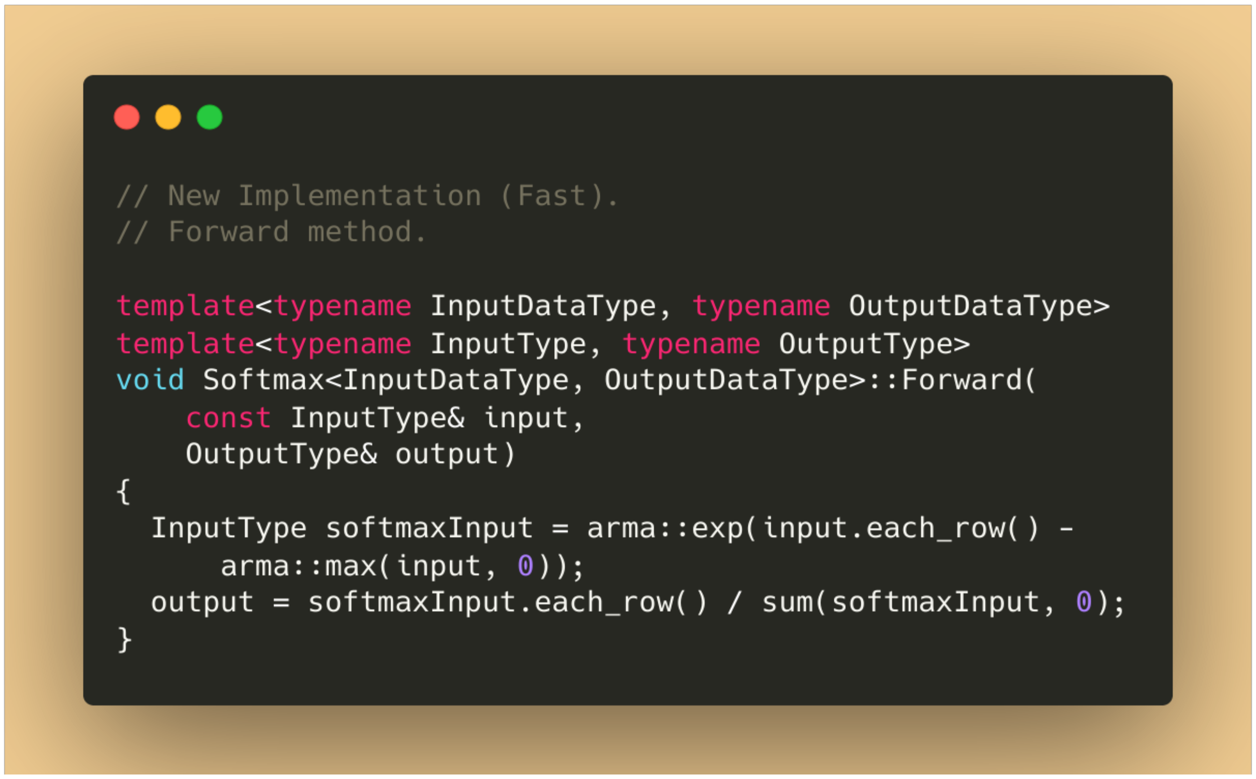


```cpp
// Forward implementation.
template<typename InputDataType, typename OutputDataType>
template<typename PredictionType, typename TargetType>
typename PredictionType::elem_type
MeanAbsolutePercentageError<InputDataType, OutputDataType>::Forward(
    const PredictionType& prediction,
    const TargetType& target)
{
  PredictionType loss = arma::abs((prediction - target) / target);
  return arma::accu(loss) * (100 / target.n_cols);
}

// Backward implementation.
template<typename InputDataType, typename OutputDataType>
template<typename PredictionType, typename TargetType, typename LossType>
void MeanAbsolutePercentageError<InputDataType, OutputDataType>::Backward(
    const PredictionType& prediction,
    const TargetType& target,
    LossType& loss)

{
  loss = (((arma::conv_to<arma::mat>::from(prediction < target) * -2) + 1)
/     target) * (100 / target.n_cols);
}

// API to call the function.
arma::mat input, target;
MeanAbsolutePercentageError<> module;

input = arma::mat("3 -0.5 2 7");
target = arma::mat("2.5 0.2 2 8");

double loss = module.Forward(input, target);
```

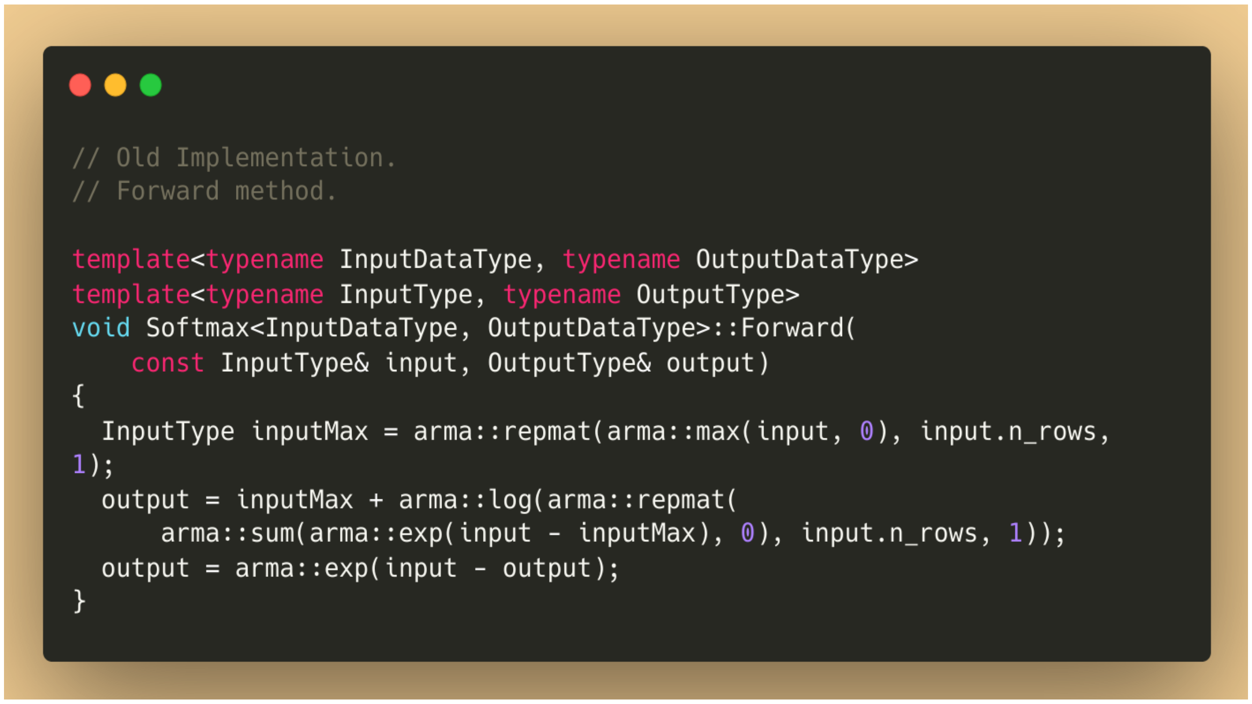
My contributions here were to add the Mean absolute percentage error in the loss functions and write tests for it to test for the edge cases such as divide by zero errors and check the correctness of the implementation, which can be seen in the PR: #2622.

- **Faster forward Implementation for softmin and softmax**

The implementation of Softmax and Softmin activation used the **repmat** function for calculation but instead using **each_row**() provided with some great performance boost and for numbers, it was a boost of approx 48%.



```cpp
// New Implementation (Fast).
// Forward method.

template<typename InputDataType, typename OutputDataType>
template<typename InputType, typename OutputType>
void Softmax<InputDataType, OutputDataType>::Forward(
    const InputType& input,
    OutputType& output)
{
  InputType softmaxInput = arma::exp(input.each_row() -
      arma::max(input, 0));
  output = softmaxInput.each_row() / sum(softmaxInput, 0);
}
```



```cpp
// Old Implementation.
// Forward method.

template<typename InputDataType, typename OutputDataType>
template<typename InputType, typename OutputType>
void Softmax<InputDataType, OutputDataType>::Forward(
    const InputType& input, OutputType& output)
{
  InputType inputMax = arma::repmat(arma::max(input, 0), input.n_rows,
1);
  output = inputMax + arma::log(arma::repmat(
      arma::sum(arma::exp(input - inputMax), 0), input.n_rows, 1));
  output = arma::exp(input - output);
}
```

My contributions in this were to try different ways to speed up the computation and measure with proper tests the average time taken by both of these implementations to decide and go for the fastest and the most accurate one, which can be seen here in PR #2657.

Summarising and mentioning Other contributions

This section summarizes all the above-mentioned and some of the other contributions that I wasn't sure if deserved much explanation or more room in the proposal.

Pull requests/ Issues	Description
PR #2620 - package name upgrade python2 to python3	This PR upgrades the package installation command for the python packages required to install mlpack bindings for python.
Issue #120 - Guidelines for adding new examples	I included some guidelines for adding new examples in the mlpack/examples repository for the newcomer that wants to contribute to the examples repository in mlpack.
Issue #117 - Error building the mnist_cnn and mnist_simple example	This issue was there because I had a previous version of ensmallen and there were no checks in the example files to ensure if an apt version of ensmallen was installed or not.
PR #124 - Ensmallen version check	To solve this I included a check in all the apt example files that checked if the minimum version of ensmallen needed is there or not and if not they throw an understandable message because when I encountered the issue as I described above it was hard for me to pinpoint that an older version of ensmallen was causing the issue.
PR #51- Fixes some spelling mistakes	This PR fixes some spelling mistakes in the README.md in mlpack/models repository.

PR #2777(mlpack/mlpack), #1(zoq/mlpack) - Swap boost:variant with vtable	This PR removes the use of boost::variant in the ANN codebase and the groundwork on this was done by me which was carried further by Marcus and then I had another PR directly to his branch that almost completes the porting process.
PR #2678, #2644, #2640, #2638, #2627 - Porting tests from boost to Catch2	In these PRs I have helped port a majority of tests from boost unit testing to Catch2.
PR #52 - Porting tests from boost to Catch2 in models repository	The models repository was still using boost unit testing so I set up the Catch2 testing there and ported all the tests from boost to Catch2 in the models repository.
PR: #2621 - Softmin Activation function	This PR adds the Softmin activation function to mlpack and completes the tests to cover the test cases and the correctness of the implementation.
PR: #2622 - Mean Absolute Percentage Error	This PR adds Mean Absolute Percentage Error as a loss function in mlpack and covers the test cases and checks the correctness of the implementations with proper and concise tests.
PR #2657 - Faster forward Implementation for softmin and softmax	The current implementation of the softmin and softmax functions were a bit slow and used repmat for some calculations and when used each_row to perform the same calculations we saw a good speed boost of 48% so this PR implements a faster softmin and softmax function.

And then there is one more thing I would like to mention in the contributions section, this was not a direct contribution to mlpack but I wrote an article on medium that introduces mlpack and covers how you can train a convolutional neural network with mlpack in c++. You can check out Convolutional Neural Network (CNN) in C++ and btw it has reached over 2.8k views by now.

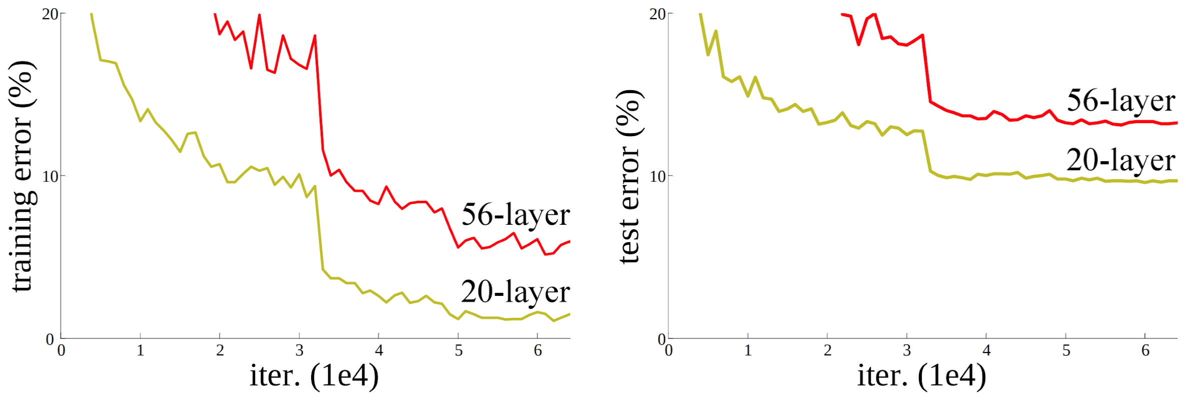
Why this specific project?

Mlpack has great support for the different type of layers in the ANN codebase and can be used to construct State of the art models and can even be used to implement the current and up to date research work but many times people want to set up some baselines or quickly want to use some model for benchmarking and project of their own and don't want to sit for hours just designing their model architecture in which they could incur a ton of errors that where this project comes in from which they can use these pre-trained/pre-defined models to quickly train on their data and compare from multiple pre-trained models that the models repository has to provide.

Insights into the new models

ResNet

Resnets first introduced in the paper Deep Residual Learning for Image Recognition was a breakthrough in training deep Artificial Neural networks. The major belief was that as you grow the number of layers in a model the accuracy and will increase and the loss/error would go down but authors in this paper experimented and showed that a Neural network with 20 layers achieved lower error on both the training and testing data whereas a neural network with 56 layers achieved poorer performance on both the datasets.

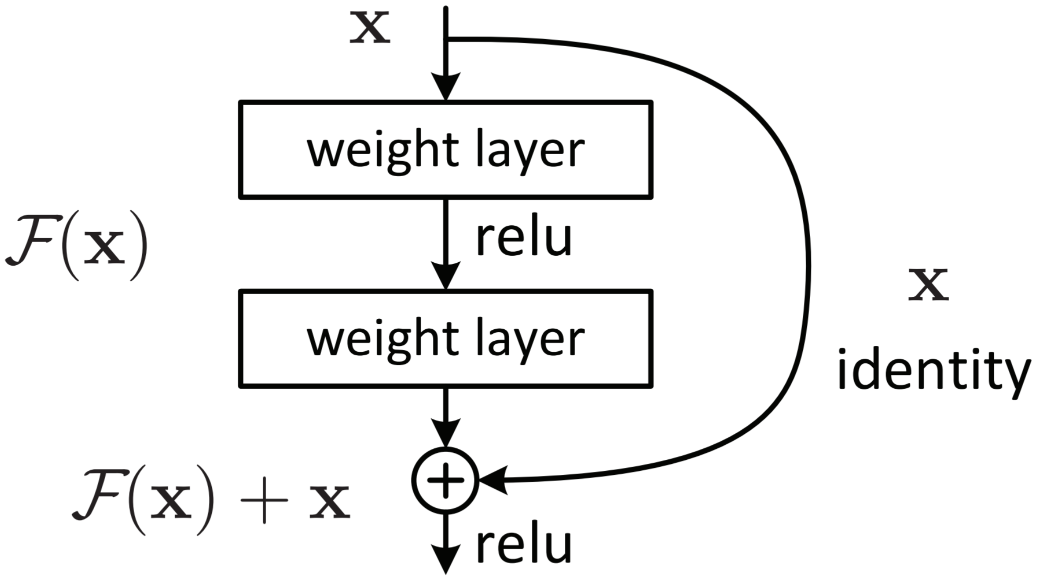


Graph Comparing training and testing error of two models (56 layers and 20 layers).

Another problem that was faced with deeper neural networks was the problem of gradient vanishing this happens when you have a deep neural network in which the layers have small gradients and so when you do backpropagation and go through multiplying all those gradients you get such a small gradient at the end that it is almost negligible for the optimizer to take any heft meaningful step and the steps taken by optimizer becomes negligible and in turn, the model can take forever to train or may not even reach the minima. The authors in the paper present residual connections which solve both of these problems and allow deeper neural networks to be trained So what are these residual connections? For this, I would be using a pictorial and formula from the paper to represent why they were superior in their time and why they are still heavily used as a benchmarking network and as a backbone to a lot of State of the art and popularly used models.

- **Residual connections**

With Residual connections, they introduce a Residual learning framework, which instead of hoping that every few stacked layers directly fit an underlying mapping it explicitly lets these layers fit a residual mapping which the other formally referred to as $H(x)$ in their paper and let the nonlinear layer for another mapping of $F(x) = H(x) - x$so the initial mapping can be realized as $H(x) = f(x) + x$, and the original mapping can be realized by a Neural network with shortcut connections.



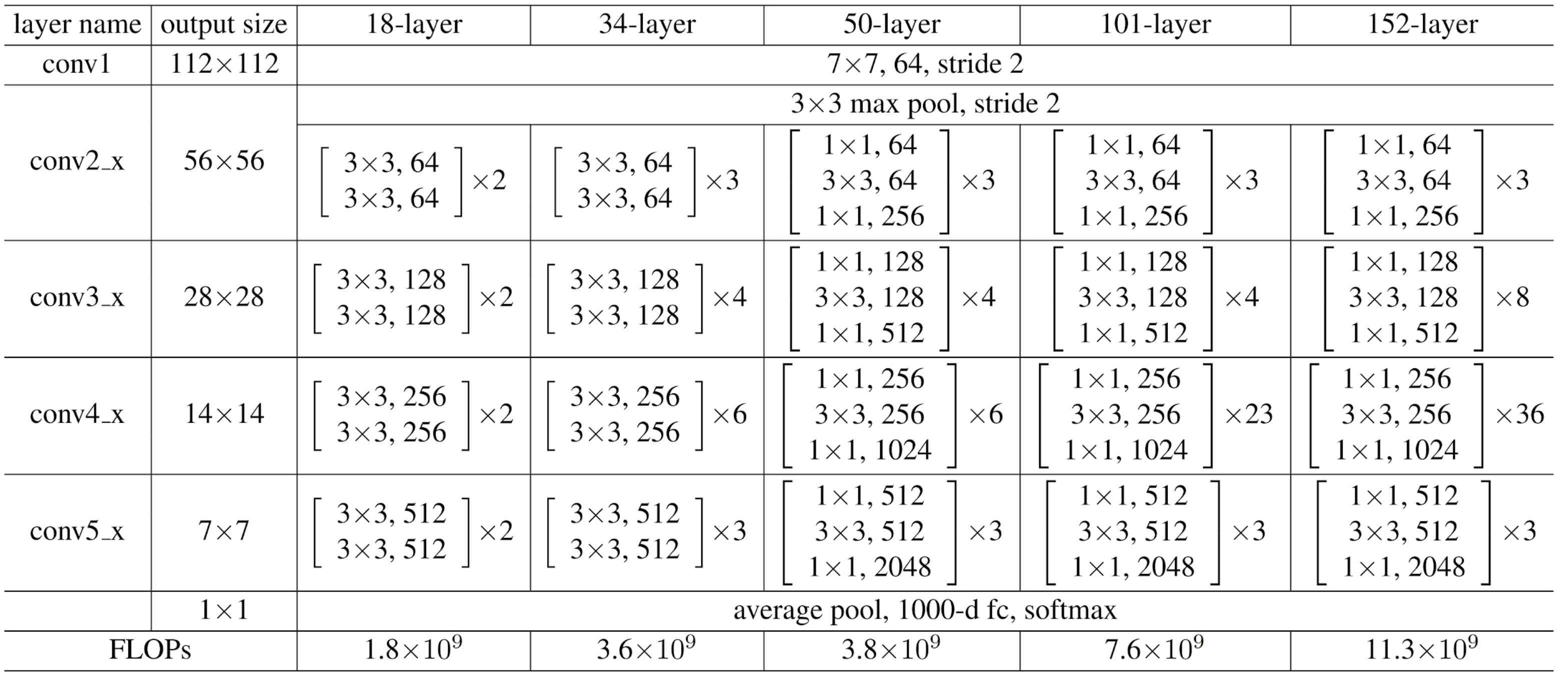
Residual Learning Block.

- **Implementation of Resnet**

The implementation of all the proposed Resnet models which are:

1. ResNet-18
2. ResNet-34
3. ResNet-50
4. ResNet-101
5. ResNet-152

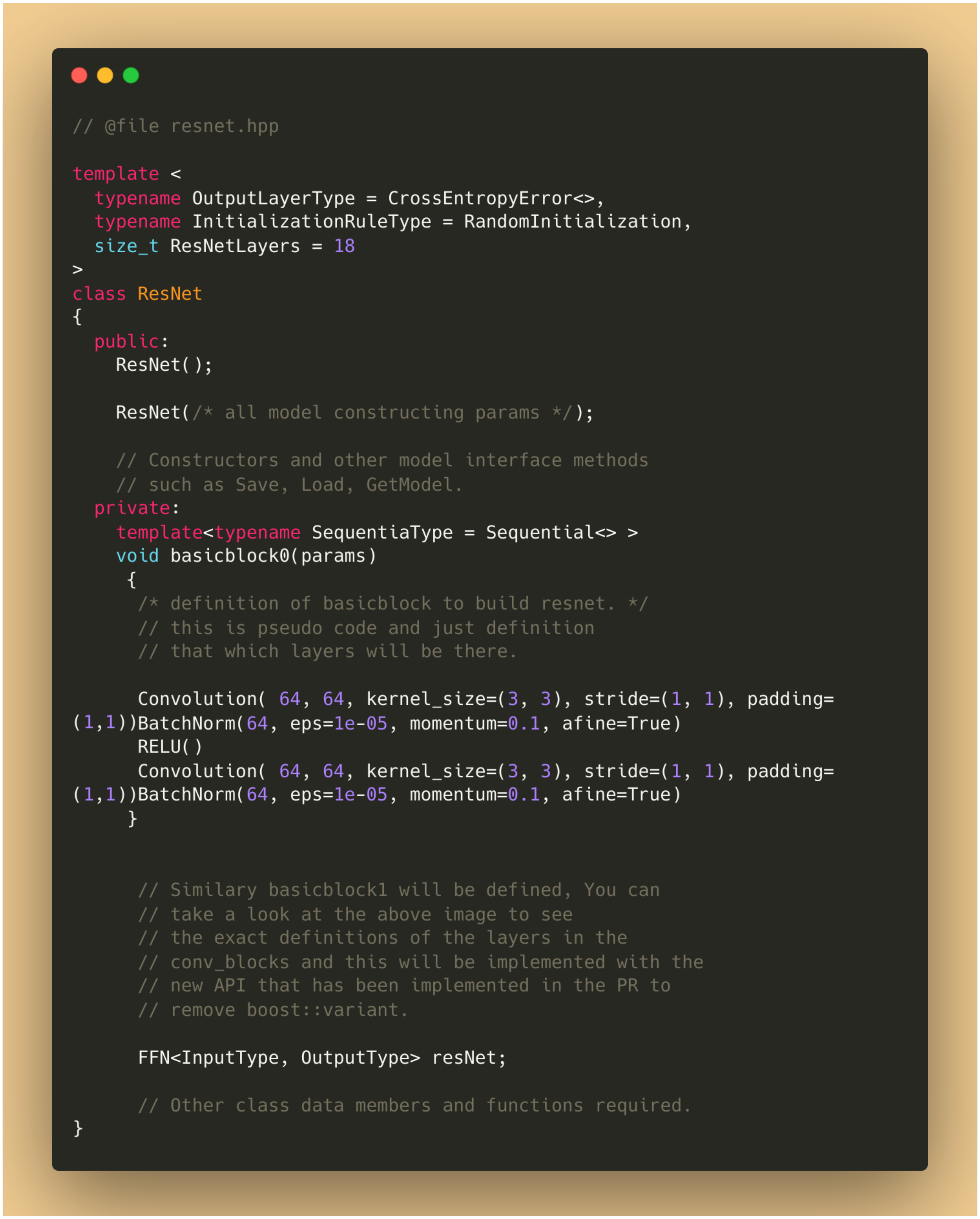
Will be done as customizable modules so we don't have to write boilerplate code again and again for each resnet architecture because internally how these models are constructed is that they have residual blocks and layers which are repeated over and over again to reach the number of layers mentioned in front of each layer but just with different layer parameters, below I have attached an image from the paper that presents all the different architectures in a tabular format and gives a more in-depth view on how they are different from each other.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Architecture for all the different Resnet models.

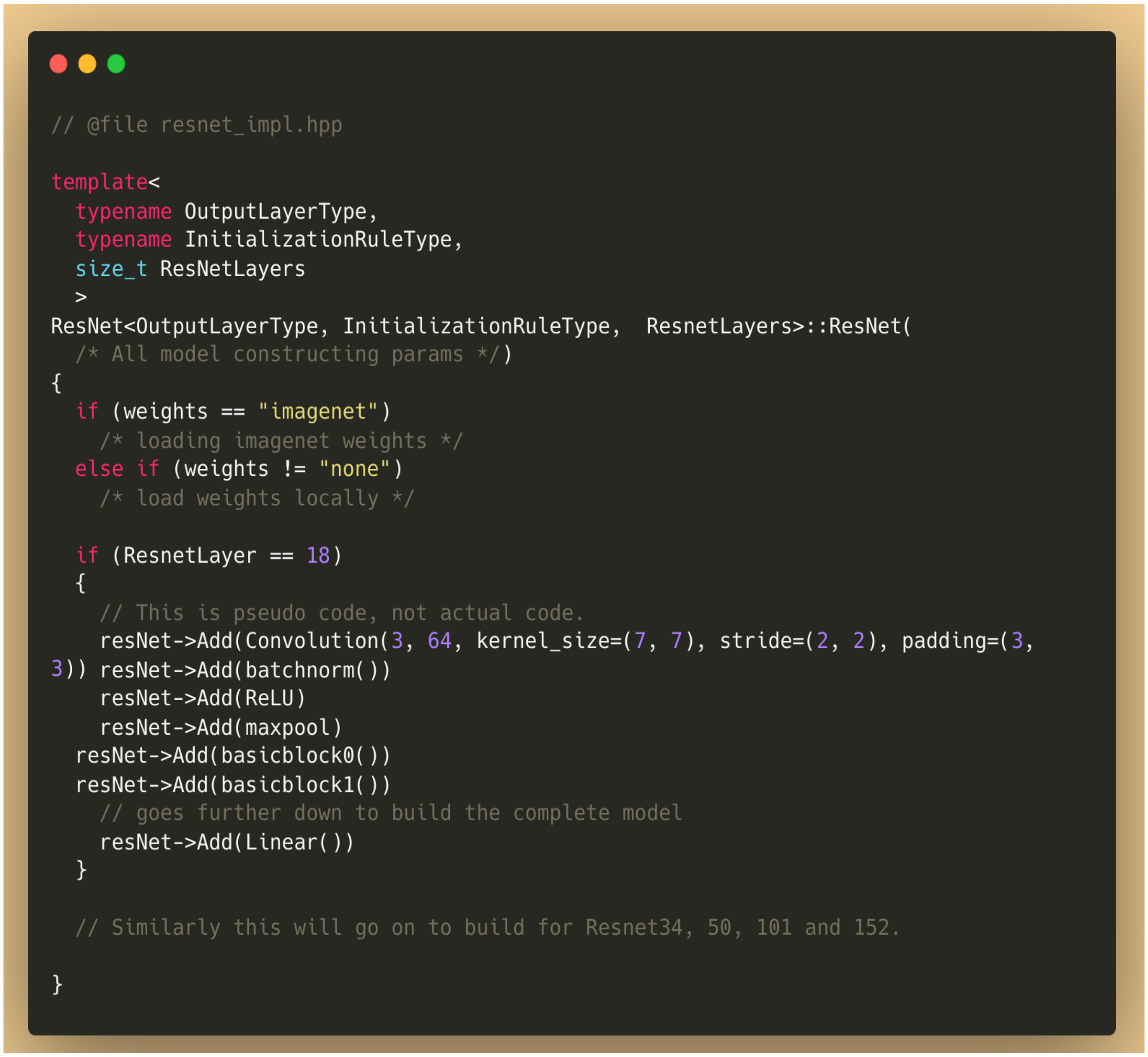
The implementation would follow a similar pattern as is followed in the models repository where there would be a resnet.hpp which would contain a Resnet class which would have all the layer blocks required to build all the different type of architectures and the resnet_impl.hpp will contain the initializations depending upon the class template parameter specified by the user for the resnet model which would be from 18, 34, 50, 101 or 152. Below I have shown some pseudo-code and not the exact implementation of the model that will be done in the GSoC period.



```
// @file resnet.hpp

template <
  typename OutputLayerType = CrossEntropyError<>,
  typename InitializationRuleType = RandomInitialization,
  size_t ResNetLayers = 18
>
class ResNet
{
  public:
    ResNet();

    ResNet(/* all model constructing params */);

    // Constructors and other model interface methods
    // such as Save, Load, GetModel.
  private:
    template<typename SequentiaType = Sequential<> >
    void basicblock0(params)
     {
      /* definition of basicblock to build resnet. */
      // this is pseudo code and just definition
      // that which layers will be there.

      Convolution( 64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1,1))BatchNorm(64, eps=1e-05, momentum=0.1, afine=True)
      RELU()
      Convolution( 64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1,1))BatchNorm(64, eps=1e-05, momentum=0.1, afine=True)
     }


      // Similary basicblock1 will be defined, You can
      // take a look at the above image to see
      // the exact definitions of the layers in the
      // conv_blocks and this will be implemented with the
      // new API that has been implemented in the PR to
      // remove boost::variant.

      FFN<InputType, OutputType> resNet;

      // Other class data members and functions required.
}
```

Corrections:

1. template<SequentialType = Sequential> in template definition of basicblock0.
2. FFN<OutputLayerType, InitializationRuleType> resNet; in the last line.



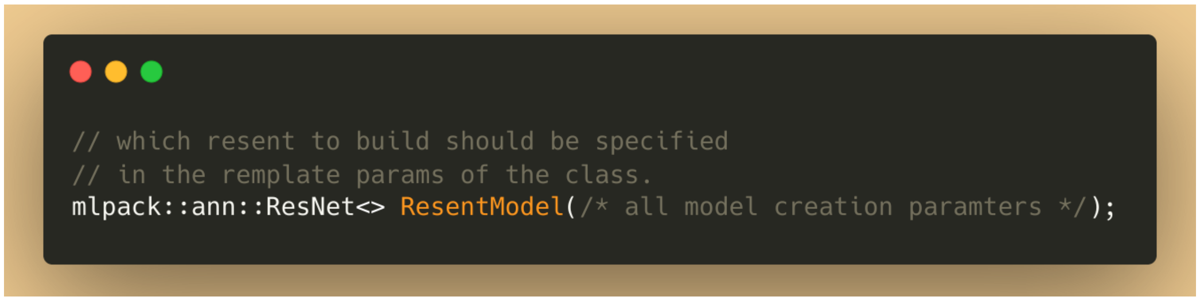
```
// @file resnet_impl.hpp

template<
  typename OutputLayerType,
  typename InitializationRuleType,
  size_t ResNetLayers
  >
ResNet<OutputLayerType, InitializationRuleType,  ResnetLayers>::ResNet(
  /* All model constructing params */)
{
  if (weights == "imagenet")
    /* loading imagenet weights */
  else if (weights != "none")
    /* load weights locally */

  if (ResnetLayer == 18)
  {
    // This is pseudo code, not actual code.
    resNet->Add(Convolution(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3,
3)) resNet->Add(batchnorm())
    resNet->Add(ReLU)
    resNet->Add(maxpool)
  resNet->Add(basicblock0())
  resNet->Add(basicblock1())
    // goes further down to build the complete model
    resNet->Add(Linear())
  }

  // Similarly this will go on to build for Resnet34, 50, 101 and 152.

}
```

Corrections:

1. `resNet.Add()` instead of `resNet->Add()`

Above I have shown some pseudo-code and a rough structure on how I will be implementing the Resnet modularly so it can construct different Resnets without much-repeated code and I will be trying to use layers that pre-exist in mlpack such as the residual layer for constructing Resnet rather than making my own but if the mlpack doesn't implement something I would do it in the model itself if it is so specific to that model and not general.

The API call to create the models will look something like this.



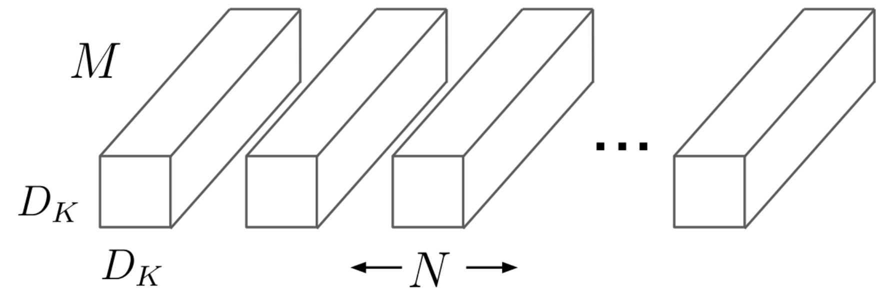
```
// which resent to build should be specified
// in the remplate params of the class.
mlpack::ann::ResNet<> ResentModel(/* all model creation paramters */);
```

MobileNetV1

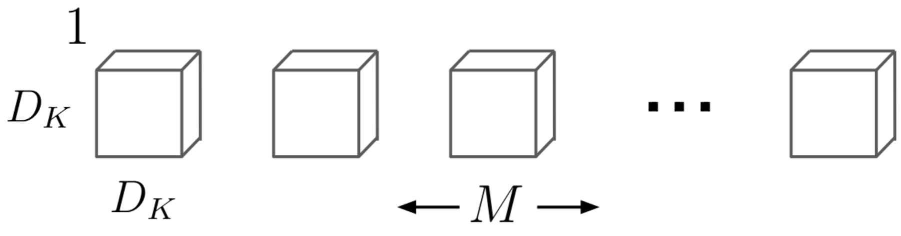
MobileNets were first introduced in the paper MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications as efficient models that were ground up designed to work on low-powered devices such as mobiles and for embedded vision applications. There were two main things to this model the first being depthwise separable convolutions and the second being two hyperparameters one which is called alpha which controls the width of the network the authors termed this as the width multiplier in the paper and if alpha has a value smaller than one it proportionally decreases the number of filters in each layer and if alpha is greater than one it increases the number of filters in each layer and if this is one the default number of filters from the paper is used. The second hyperparameter being depth multiplier which was termed as resolution multiplier in the mobilenet paper, this is applied to the input and the internal representations of every layer to significantly reduce the computational cost.

- **Depthwise separable convolutions**

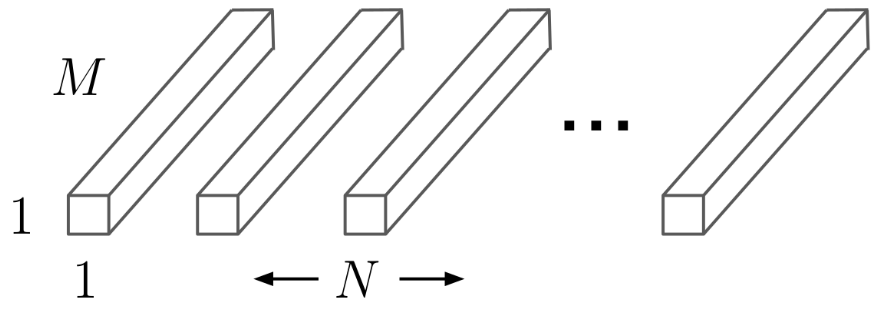
These are a form of factorized convolutions that factorize a standard convolution into depthwise convolution and a 1 * 1 convolution called a pointwise convolution. For mobileNet, the depthwise convolution applies a single filter to each input channel and the pointwise convolution then applies a 1 * 1 a convolution to combine the outputs produced by the depthwise convolution. Below I am attaching an image to display the difference between standard convolution filters and depthwise separable convolution filters and I am trying to not go into the details of how they work and how they reduce the computation those are some of the things that can be easily understood by the paper I would like to focus on the implementations here.



Standard Convolution Filters



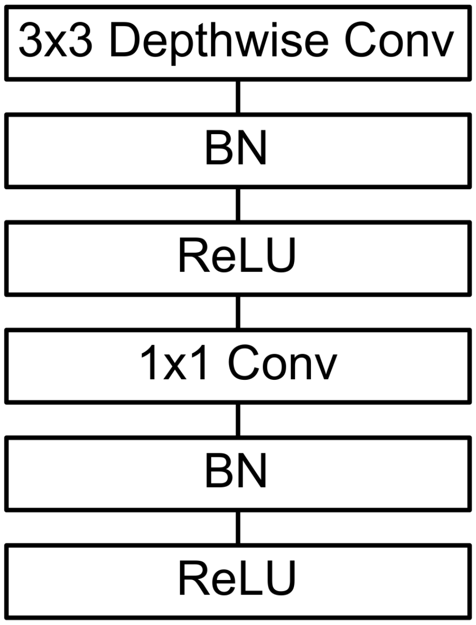
Depthwise Convolutional Filters



1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

- **Implementation of Depthwise separable convolutions**

In the Implementation for Depthwise separable convolution, the block contains batchnorm and ReLU after both the depthwise convolutions and 1 * 1 pointwise convolution as shown in the diagram below.

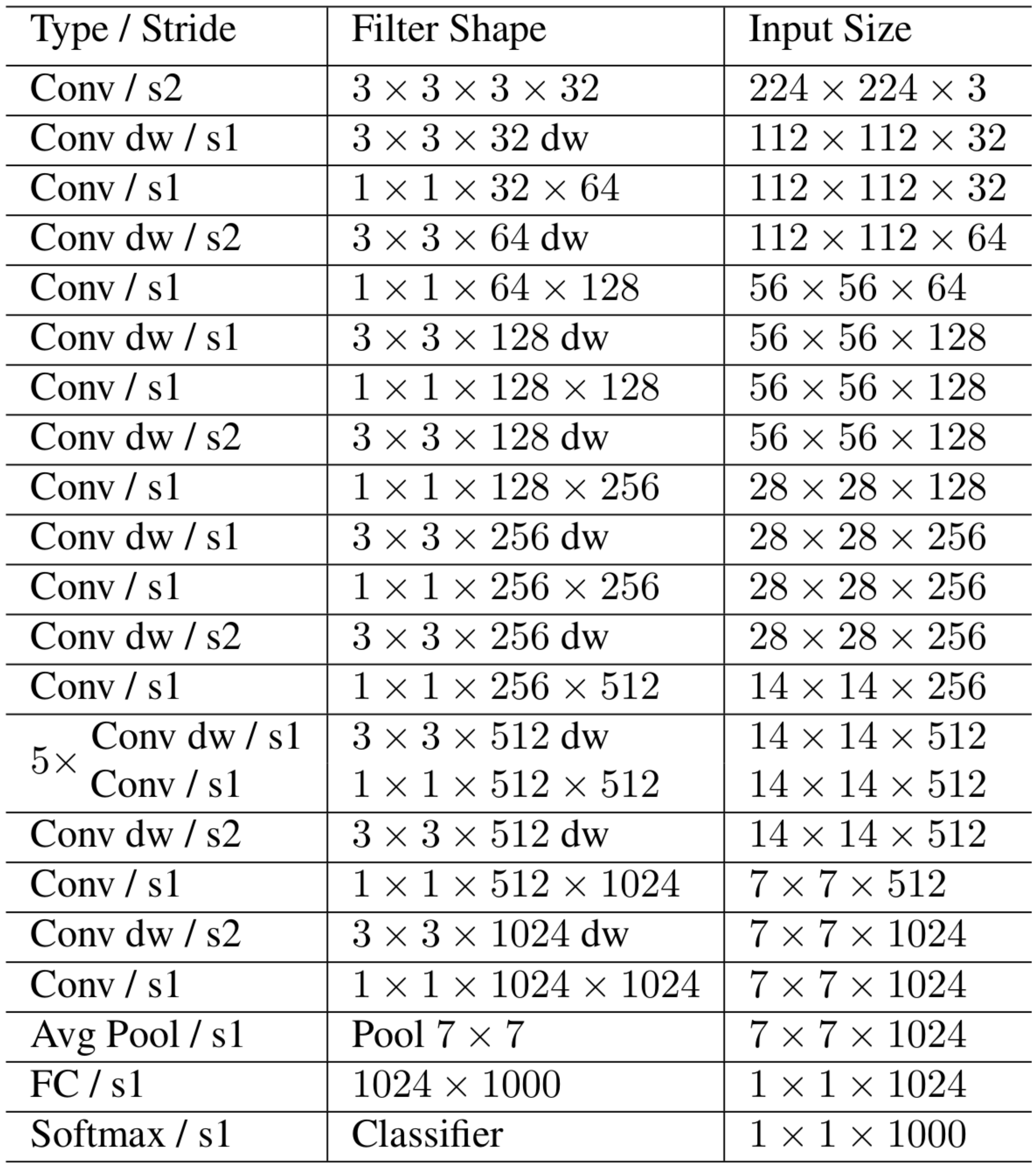


Depthwise Separable convolutions with Depthwise
and Pointwise layers followed by batchnorm and ReLU.

A good amount of work has already been done to add depthwise separable convolutions in Addition of Depth-wise Separable Convolution (Separable Convolution) Layer so I will be taking it further from there and complete the PR with all the changes and get it merged in the mlpack codebase so it can be used to construct MobileNetV1.

- **Implementation of MobileNetV1**

Below I will present an Image for the architecture of MobileNetV1 that should clear out most of the doubts about which and what kind of layers are used to build the model and for now the MobileNetV1 model will only be predefined but if the situations are favorable I will try to add the weights for the default MobileNetV1 by which I mean where the input size is 224 * 224 * 3 and alpha is equal to one which will be trained on the ImageNet dataset.

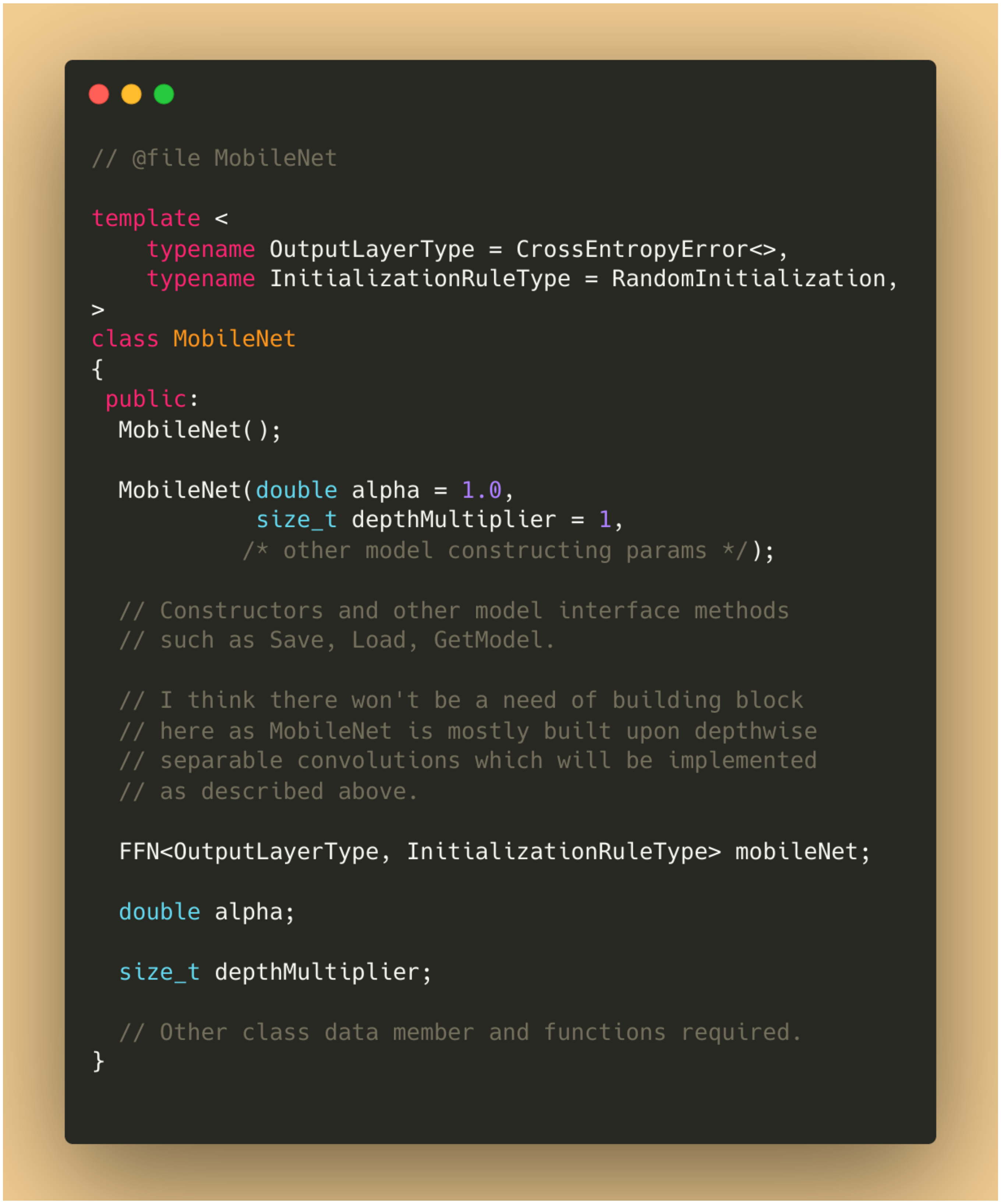


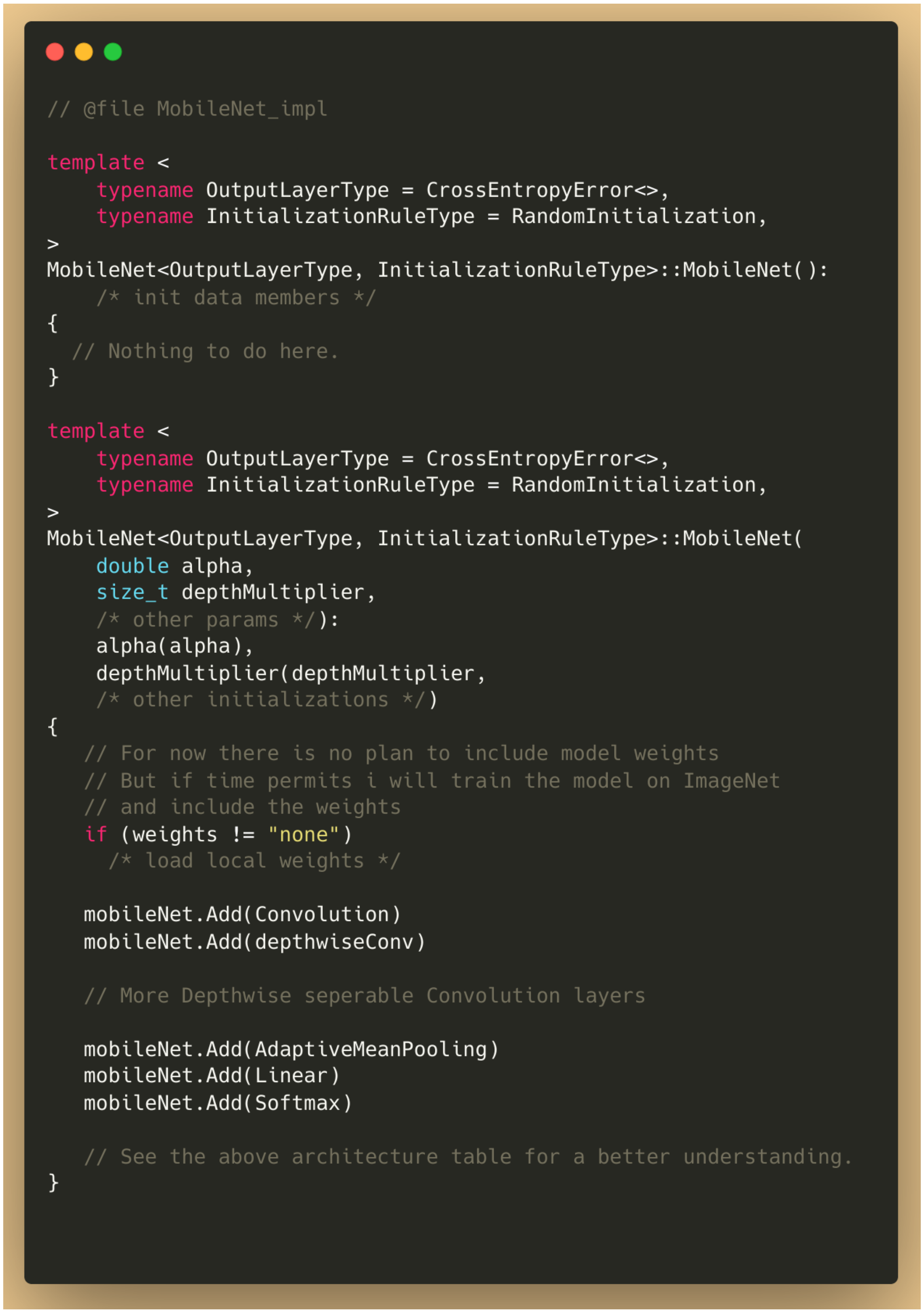
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet Body Architecture. Conv dw is depthwise convolution and just the layer below those where the first dimensions are 1 * 1 are pointwise convolutions(Conv / S1).

The implementation of the MobileNetV1 will allow the user to construct mobileNetV1 with different alpha and depth multipliers which affect the model architecture. Below I am presenting the pseudo-code on how this will be implemented in the models repository.

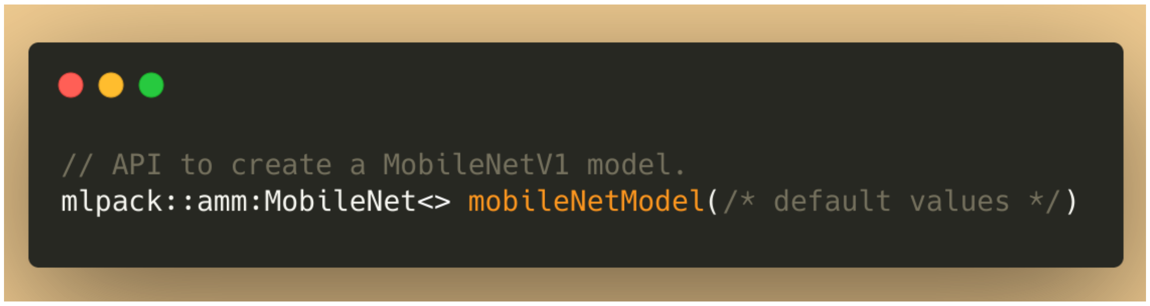
Also, I would like to mention that in the codebase MobileNetV1 would be called MobileNet because I have a further intention to add MobileNetV2 and MobileNetV3 if I am done with my proposed deliverables or after GSoC ends.


```cpp
// @file MobileNet

template <
    typename OutputLayerType = CrossEntropyError<>,
    typename InitializationRuleType = RandomInitialization,
>
class MobileNet
{
 public:
  MobileNet();

  MobileNet(double alpha = 1.0,
            size_t depthMultiplier = 1,
           /* other model constructing params */);

  // Constructors and other model interface methods
  // such as Save, Load, GetModel.

  // I think there won't be a need of building block
  // here as MobileNet is mostly built upon depthwise
  // separable convolutions which will be implemented
  // as described above.

  FFN<OutputLayerType, InitializationRuleType> mobileNet;

  double alpha;

  size_t depthMultiplier;

  // Other class data member and functions required.
}
```



```cpp
// @file MobileNet_impl

template <
    typename OutputLayerType = CrossEntropyError<>,
    typename InitializationRuleType = RandomInitialization,
>
MobileNet<OutputLayerType, InitializationRuleType>::MobileNet():
    /* init data members */
{
  // Nothing to do here.
}

template <
    typename OutputLayerType = CrossEntropyError<>,
    typename InitializationRuleType = RandomInitialization,
>
MobileNet<OutputLayerType, InitializationRuleType>::MobileNet(
    double alpha,
    size_t depthMultiplier,
    /* other params */):
    alpha(alpha),
    depthMultiplier(depthMultiplier,
    /* other initializations */)
{
   // For now there is no plan to include model weights
   // But if time permits i will train the model on ImageNet
   // and include the weights
   if (weights != "none")
     /* load local weights */

   mobileNet.Add(Convolution)
   mobileNet.Add(depthwiseConv)

   // More Depthwise seperable Convolution layers

   mobileNet.Add(AdaptiveMeanPooling)
   mobileNet.Add(Linear)
   mobileNet.Add(Softmax)

   // See the above architecture table for a better understanding.
}
```

The API to create a MobileNet model will look like this.



```cpp
// API to create a MobileNetV1 model.
mlpack::amm:MobileNet<> mobileNetModel(/* default values */)
```

The above-described models contain a lot of layers and a lot of parameters that can go wrong and so they will be tested with Catch2 for all the different variants of models they can create and for now the plan is to use the [mlpack PyTorch weight translator]() to extract weights for resnet from Pytorch and so the dataset automatically becomes ImageNet for resnets and MobileNet will only be predefined for now.

Timeline

May 17 - June 7	This will be the community bonding period. I would like to utilize this time to get the PR to remove boost::variants merged and get the models repository up to the same level as the mlpack repository. We know mlpack has seen some major code porting and keeping the model repository in sync with mlpack will ensure a smooth transition of the models repository into the mlpack package itself because I feel like that is something that can happen in the future and would be great to ship with ready to build model support with the package itself. I have just mentioned one or two things the models repository is lacking in but other things come up as I go to sync them up.
Below I will describe the timeline for the coding period that is from June 7 to August 16, I will not go into much detail because I have explained the majority of my project above and as to what I will do in each week, that is something that might change in 10 weeks but I will describe what I would like to work on each week and the tests would be written after I complete writing one model and not in between and after that I would take time to document these	

examples on the wiki page for models repository too. So there totally will be three things that I will not mention in every line below: Implementation, testing, writing examples, documentation, and updating any references maintained.	
June 7 - June 27	Complete Resnet module
June 28 - July 11	Testing and documentation of the resnet module.
July 12 - August 1	Complete MobileNetV1 module
August 2 - August 15	Testing and documentation of the MobileNetV1 module.
August 16 - August 22	Adding an extra buffer week to compensate for any irregularities that I may encounter in the above split.

By now I should have completed the implementation of the Resnet and MobilenetV1 module and then I would like to add MobileNetv2 and MobileNetv3 which would in the future allow me to add DeepLabV3 as a segmentation model and to accompany that I would also like to add a Segmentation data loader that supports COCO format.

Post GSoC Contributions.

And after the GSoC period ends, I would still like to implement the things that I mentioned above, and being a part of the mlpack community I have gained the knowledge of codebase that will help me make more contributions in the future and I would be glad to guide the upcoming students as a mentor next year.

Blogs

I would like to document the work I do every week which will help me with a couple of things, first, it will keep a check on the timeline that I proposed and would document the issues and experiences that I had every week and I think will also help my mentor evaluate the complete work, problem and how I utilize my time in that specific week.

Conclusions

I have been working with mlpack and contributing towards it since October 2020 and I have loved the feedback and responses I get on the ideas, pull requests, and all the issues and I will be thrilled to work with mlpack for GSoC.

I tried to explain everything in as much detail as possible and I do realize this application got long so thank you so much for reading till here.