EE569 Digital Image Processing

HOMEWORK # 5

Aagam Shah

USC ID-8791018480

USC Email – aagamman@usc.edu

Submission Date – April 7, 2020

Problem 1: CNN Training on LeNet-5

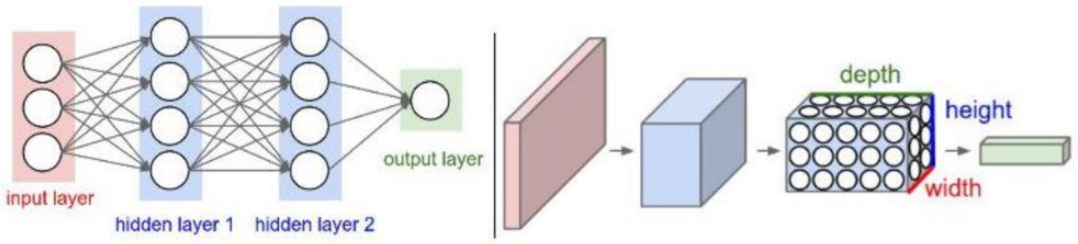
(a) CNN Architecture:

1. ABSTRACT & MOTIVATION

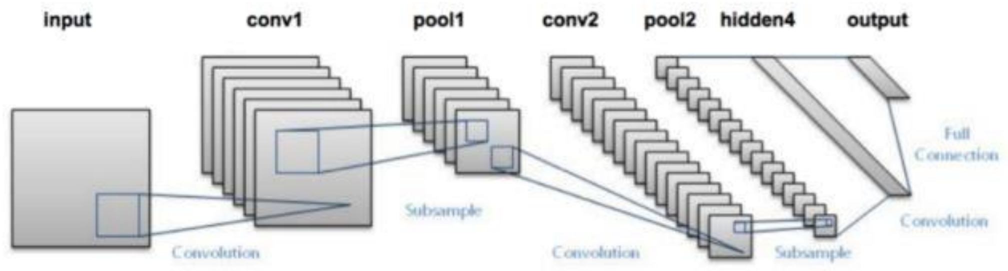
Artificial Neural Networks (ANN) plays a very important role in the field of Artificial Intelligence whose idea is basically inspired by the biological neural networks which is like the neurons of an animal brain. An Artificial neuron in basically the connection of nodes known as artificial neurons like the neurons of the animal brain. Each connection similar in the animal brain transmit a signal to the other neurons. So, an artificial neuron which will receive the signal will then process it and transmit the information to the neurons connected to it. So, here the Convolutional Neural Network (CNN) is the feed-forward type of Artificial Neural Network (ANN) where the connection pattern of the neuron is similar to the neurons of the visual cortex of an animal brain and the signals are transmitted only in one direction that is the forward direction as suggested by the name. Convolutional Neural Network is nothing but a multilayer perceptron which has fully connected networks, i.e., every neuron in each layer is connected to every other neuron in the next layer.

A Convolutional Neural Network (CNN) is basically a part of Deep Neural Networks, a category of Deep Learning. CNN's are more commonly found in Visual Imagery applications such as Image and Video Recognition, Image Analysis and Classification, Face Recognition, Computer or Robot Vision, Self-Driving Cars, Security, Medical Imaging, AR-VR applications, etc.

The term Convolutional Neural Network implies that a mathematical operation known as convolution is employed by a certain network. So, in simple terms if we say that the Convolutional Neural Network (CNN) are basically the neural networks which will use the mathematical operation known as Convolution, replacing it with the normal matrix multiplication in maximum of one layer of the Neural Network.



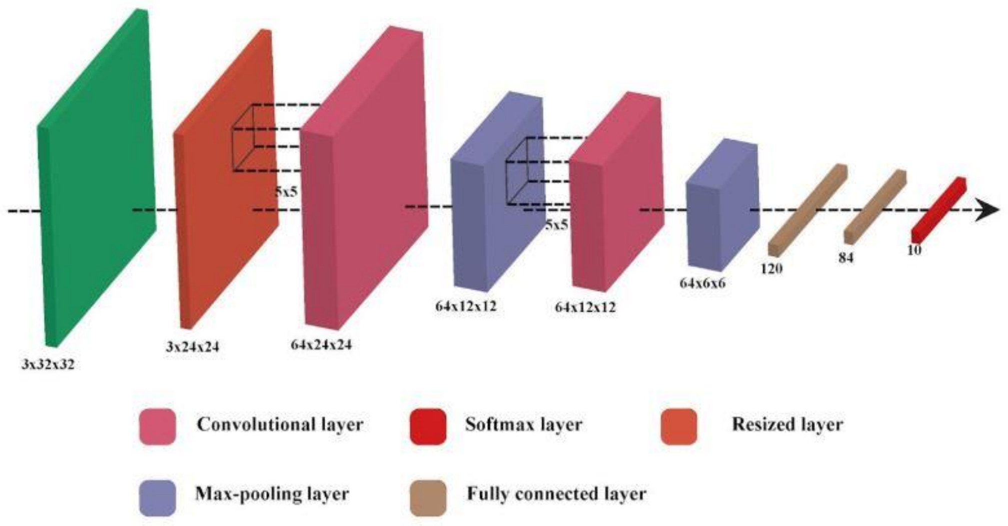
Convolutional Neural Network is technically a self-learning process which means that it extracts the features and then will classify them automatically, which in turn results in less classification time but computationally intensive.
Since CNN can be computationally intensive, so for big dataset and heavy tasks, in practice all such kind of machine learning activities are carried out using high performance GPUs and multi core processors. But here as we have relatively smaller dataset of CIFAR-10 which contains 50,000 training data images and 10,000 testing data images where each train as well as test images are of 32x32 dimension, so I can carry out this machine learning task on my system only. So, in this problem I am required to train a simple convolutional neural network called **LeNet-5** invented by Yann LeCun the director of Facebook AI research which was basically designed for handwritten and machine printed character recognition. Since we know that the LeNet-5 model of CNN architecture was designed mainly for character recognition and was not meant for image recognition or image classification so it is obviously expected to have lower accuracy to be obtained at last using this architecture for image recognition or image classification as compared to the accuracy of the LeNet-5 model used for character recognition.



2. DISCUSSION

Answer 1:

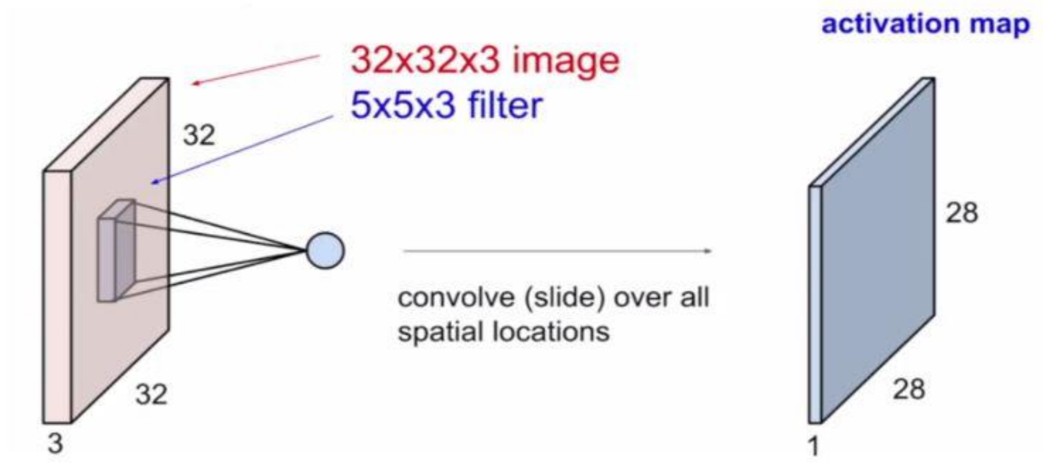
The Convolutional Neural Network (CNN) basically consists of an input layer and an output layer along-with the multiple hidden layers in between the input layer and the output layer. These hidden layers of the CNN architecture basically consist of Convolutional Layers, Max Pooling Layers, Fully Connected Layers and Normalization Layers.



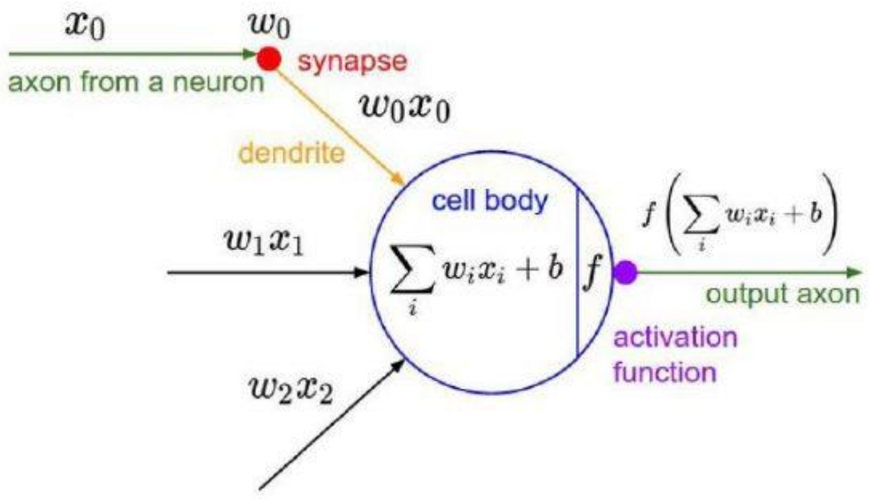
The architecture and the operational mechanism of the convolutional neural network is as follows by describing the CNN components as:

(i) **<u>The Convolutional Layer:</u>**

The convolutional layer is basically the most prominent layer of CNN architecture. Although it is said to be a convolutional operation, but technically it is nothing but mathematically a sliding dot product or in other words it can be said as cross-correlation. This layer in the entire CNN architecture performs the most computationally heavy calculations. The input to this convolutional layer is a tensor having the shape as (width x height x depth (number of channels)) x number of input images. This input tensor is then passed through a set of the learnable filters of the convolutional layer which are spatially small but reaches till the depth of an entire input image. So, in detail the input tensor image is passed in the forward direction in the feed forward network by performing convolution of each filter with the input image along the height and the width of an entire input image, which is nothing but technically computing the sliding dot product of the filter elements and the pixel elements of an input image at each and every position. In the process of performing this operation of 2D convolution the input image is abstracted as a feature map in the response of filter in each spatial location each feature map having the dimensions as (width x height x depth (number of channels)) x number of images, which means that depending on the total number of convolutional layers they are stacked in 3 dimension to generate an output volume of images.



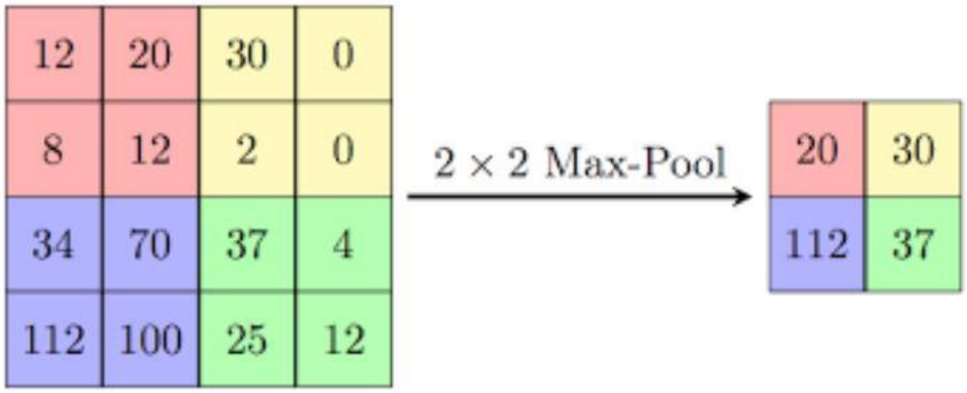
In simple terms, let us consider the volume of an input layer to be H1 x W1 x D1 (height x width x depth) and the volume of an output layer to be H2 x W2 x D2 (height x width x depth). Now, a 5 x 5 kernel of local receptive filter is used to scan over an entire input image both along the height and the width in order to obtain the similarity of features across the image. In this way depending upon the total number of local receptive filters we generate the different feature maps for an input image which when stacked together will form a stack of output layer each having different extracted features. Basically, these receptive filters are used to extract different features such as an edge, corners, textures, etc.
The Convolutional operation can be defined as the cross-correlation operation. Pictorially, it can be represented as:



Here, as we can see that the bias term is added to the inner product of the image patch and the filter weights as $\sum_i w(i)\, x(i)$.
The activation of a neuron is determined by a non-linear function (f).

(ii) **The Max Pooling Layer:**

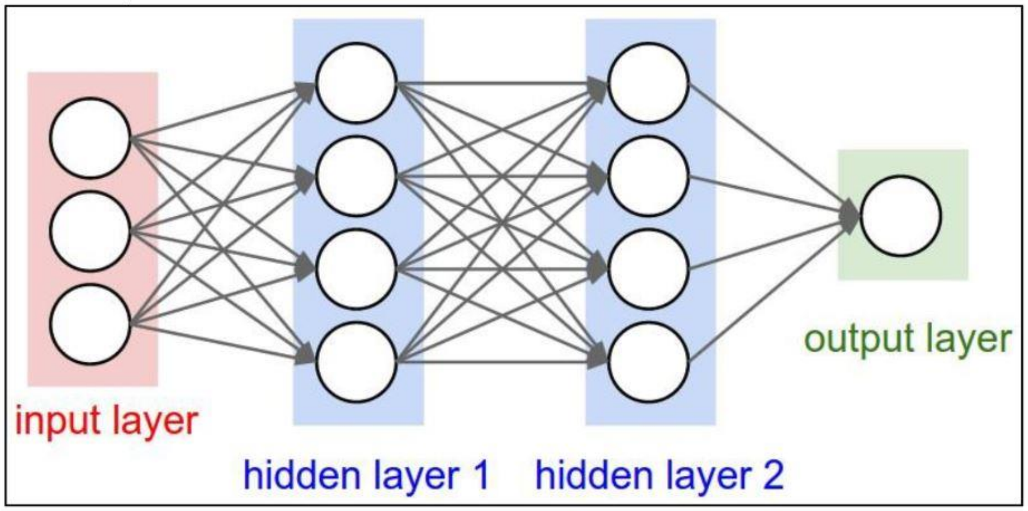
The main function of this layer is to streamline the computation either locally or globally for which it may have either local or global pooling layers. The function of the local pooling layer is to combine the small clusters (2 x 2) while the function of the global pooling layer is that it will basically act on all the neurons of the preceding convolutional layer.
This layer is basically located in succession to the Convolutional layer. As we know that the feature detection has more priority as compared to that of the exact location detection because the localization of patterns is not that important as they can be located at any position of an image. So, as a result the relative position plays a more prominent role. So, this step basically demands the feature reduction or decrease in the variations in the identification of an object which leads to the reduction in location variance, thereby variation distortion and the shift from output are decreased.
Pooling layer progressively reduces the spatial dimension of the output image produced by the preceding convolutional layer the parameter reduction or by combining the outputs of clusters of neurons in one layer into a single neuron in the next layer.
Here, we perform the local pooling operation which are also two types namely: Average or Max Pooling. Basically, in Average pooling we take the average value from each cluster of neuron present in the preceding layer whereas in the Max Pooling we take the maximum value from each cluster of neuron present in the preceding layer.



The Pooling operation is performed independently on every depth slice of an output stack of feature maps from the convolutional layer which act as input for the pooling layer. The pooling layer commonly uses 2 x 2 sized filters to down sample every feature map of input by a factor of 2 along the height as well as along the width. But the depth dimension in the pooling operation remains unchanged. This technique helps in 50% reduction in the size of the feature map and 75% of the activations are discarded thereby reduction in the computations for the further processing steps.

(iii) **<u>The Fully Connected Layer:</u>**

The Fully Connected Layer, by the name itself justifies that every neuron in the output of this layer is connected to all the neurons in the previous layer which is activated by the non-linear activation function which is based on the principle of the multi-layer perceptron. Fully connected layer basically stretches out the input volume to map it against N dimensional output vector.

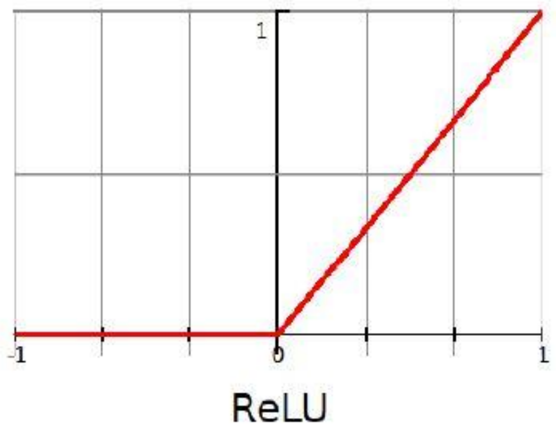


In simple terms, this layer consists of all the connections with all the activations of the previous layer. The activation function is calculated by the means of matrix multiplication with a bias offset. The input to this layer is fed as the input volume from the previous layer which can be either from the Convolutional Layer or the Pooling Layer. The output obtained from this Fully Connected Layer is nothing but a N dimensional vector, where N can be chosen according to the number of classes required as the problem statement. So, basically the this layer is generated by taking into consideration the output of the previous layer which contains the features maps of the high level features and then detect the features which closely correlate to a particular class having weights such that when we compute the product of weights and the preceding layer, we can hence achieve the true probabilities for separate classes.

(iv) **<u>The Activation Function:</u>**

The main aim of the Activation function is to create the non-linearity of the neural network, i.e., to achieve the non-linear decision boundary. This is because the amalgamation of the weights and the biases will give linear outputs and hence linear decision boundary make not work well for kind of data types so it’s reasonable to use the non-linear decision boundary in order to classify the data, which is achieved with the help of the Activation Function.
The Activation function basically defines the set of outputs for a given set of inputs. The operation of an Activation Function is that an artificial neuron computes the weighted sum of inputs and add bias which in turn will give an output

based on which the activation function decides that whether the neuron should be kept or fired. Thus, the activation function basically helps to introduce the nonlinearity in the network because the combination of linear weights and bias layers with the help of linear activation layer will eventually yield a linear function, so for inculcation of non-linearity in the network the nonlinear activation function are used.
In simpler terms, the activation function is nothing but a node which is placed in between or at the end of the network. The prominent role of the Activation function is to transform the input signal from linear to nonlinear nature and then based on the decision rule the nonlinear activation function will decide whether to keep the neuron or not. There are different types of activation functions such as Sigmoid, Tanh, ReLU, Leaky ReLU, etc. but the most commonly used Activation Function is the ReLU function (Rectified Linear Unit) because of the fact that all the neurons are not activated at the same time.



(v) **<u>The SoftMax Layer:</u>**

Mathematically, the SoftMax Function is defined as follows:

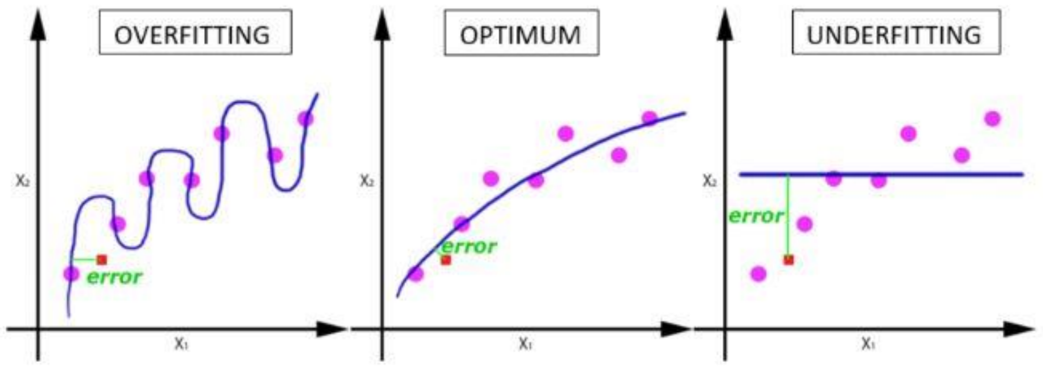
$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Where *z* is a k-dimensional normalized probability vector which will return either value close to 0 or 1. This layer is nothing but the generalization of the logistic regression. The main function of this layer is learning the parameters from the model and converge faster.
This is basically the final 1D layer of the neural network. So, the SoftMax Layer should have the same amount of nodes as that of the output layer, i.e., this layer will have 10 classes each class representing one element. The input to the SoftMax layer is the N dimensional vector from the Fully Connected Layer, which then transforms it to a real number ranging from 0 to 1 which in turn will add up-to 1

like the data put into pdf. The distribution of probabilities aids in the input classification.

Answer 2:

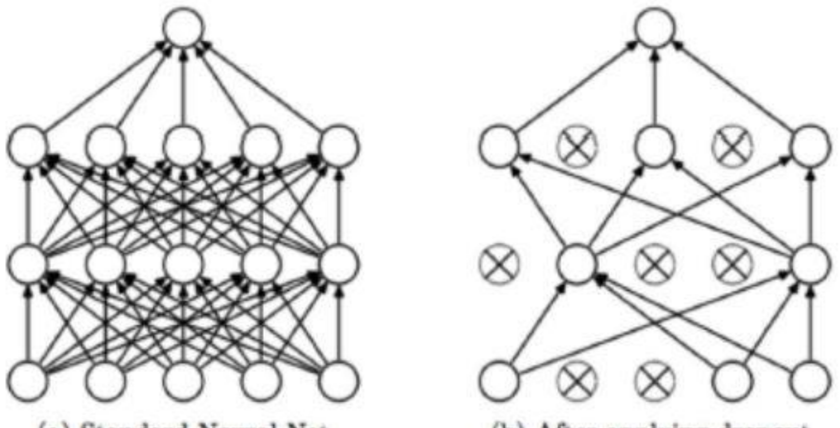
Overfitting issue in model learning implies the situation that when the trained neural network over matches the training dataset and hence the ability of generalization is lost. Overfitting issue in the model learning is the phenomenon in which that if the input dataset is changed then the difference between the expected and the predicted value will be very huge thereby creating a large performance gap between the training and the testing accuracy with training data having the best performance and will perform poor for the next set of samples. This phenomenon generally occurs when the model becomes too good or fits too well for the training dataset. In simple terms it means that the model has almost memorized the training data and its curve fitting for the training data samples and has in short generalized the model thereby reluctant to get adapted to the new input dataset. This in turn leads to the higher accuracy of the training dataset along-with the declination of accuracy of the testing dataset and hence its unable to fit the testing data unless it resembles close to the training dataset. Overfitting generally occurs when the model parameters are tuned very finely to minute level in order to obtain the best training accuracy. Also, when the number of training samples are smaller than the number of the training parameters, leads to the overfitting conditions.



The techniques that can be used in CNN training to avoid the overfitting conditions can be listed as follows:

a. **Regularization:** This technique indicates that the overfitting can be avoided by making the network model less complicated which can be done with addition of the regularization/penalty term to the objective equation/cost function in order to minimize the error between the expected value and the predicted value thereby making the network model simple, thereby adding L1 and L2 norm layers.
b. **Data Augmentation:** In this approach the preceding knowledge is utilized in order to add more amount of training data which leads to the increase in variety of data in the form of geometric modifications, inculcation of noise, distortions, rotations, etc. can be performed to generate larger dataset of the training data.

c. **Dropout:** In this approach a dropout layer is added in the Convolutional neural network generally after the first fully connected layer to prevent the overfitting issue. This is one of the most well-known technique applied in the deep learning neural networks. The basic function of this layer is that it eliminates the arbitrary number of the initiations by deactivating them i.e., bringing them down to 0 at every update such that in the successive iterations of training the weights aren't updated as the training samples are much lower as compared to that of the amount of training parameters. This may technique may lead to the loss of data but more importantly helps in avoiding overfitting issue. So, as a result it is advisable to use a low dropout value in order to prevent too much loss of data and at the same time to avoid the overfitting issue.


(a) Standard Neural Net (b) After applying dropout.

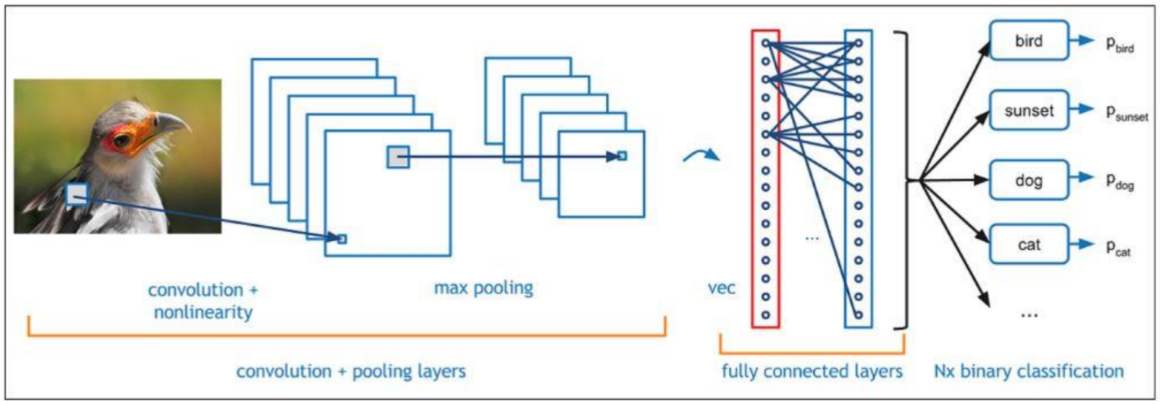
d. **Cross-validation:** This approach is based on the fact that the training dataset is bifurcated into the training samples and the validation set at random for every iteration to fit the model. Then following it the fine tuning of model parameters is done in order to fit the training samples and the they are tested on the validation set. Thus, the random behavior of the training samples is inhibited, and the generalized data of validation set is obtained.

e. The easiest way of avoiding overfitting issue is to prevent to many iterations during the training process which can be achieved by stopping the iterating process once the accuracy stops changing. This technique is known as **Early Stopping**.

Answer 3:

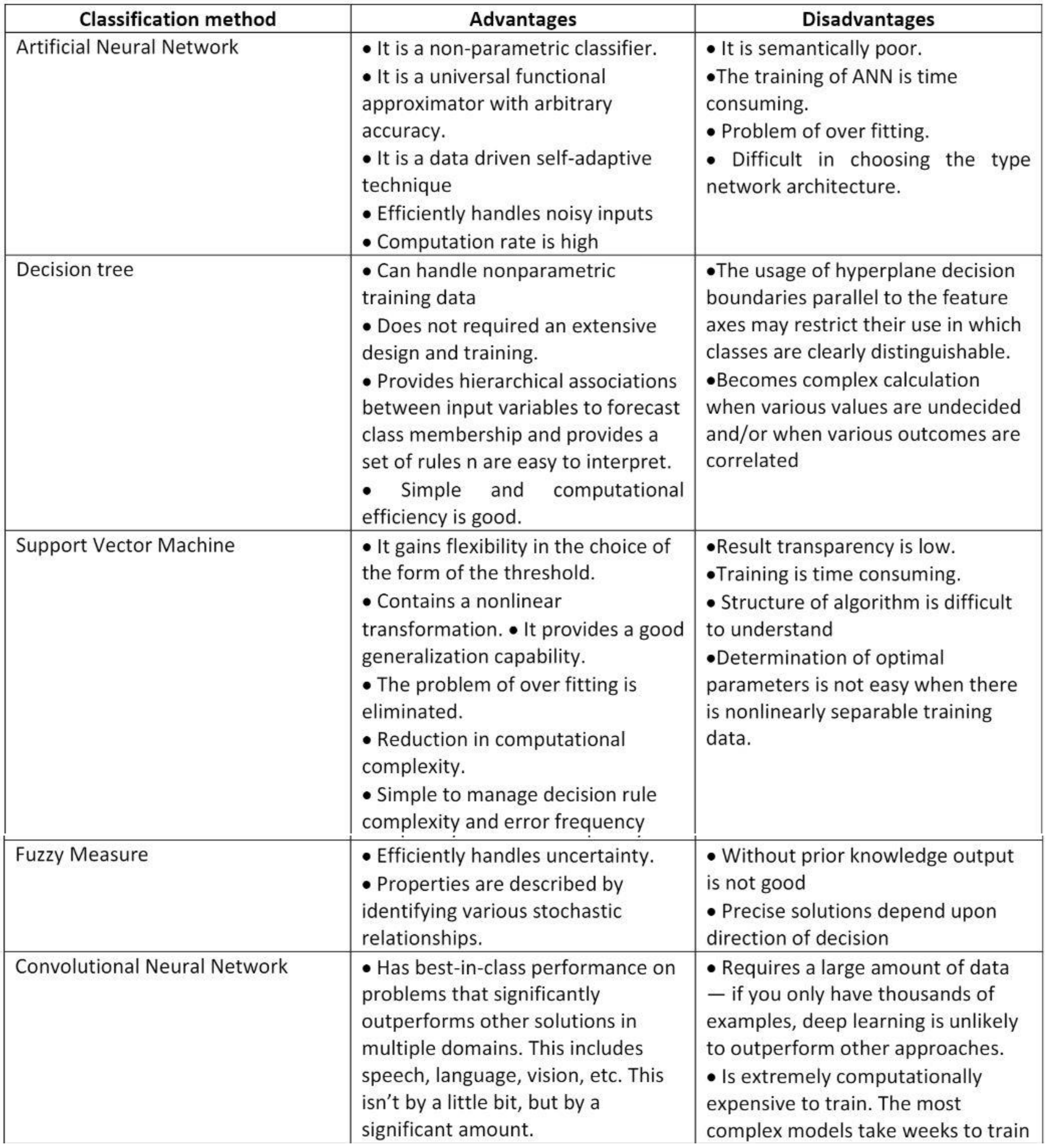
CNNs work much better than the other traditional methods in many computer vision problems because of the fact that the traditional neural networks are fully connected which in turn makes it computationally rigorous thereby making the network complex to get trained, whereas on the other hand using CNN this can be avoided with the help of local connection and the sharing of weights to prevent the complexity in the feature extraction, reconstruction and classification process.

Also, CNNs performs much better than the traditional methods of the computer vision because of the fact the CNN is data driven approach, which means that the filters learn themselves automatically from the labeled data. For normal image classification procedure, in order to classify an object, the filters which can match to the corresponding classes are required. So, in the traditional methods the humans designed these filters, but it becomes very difficult to design the filters for the complex applications and hence may not be capable to achieve the underlying pattern of an object clearly thereby leading to a poor accuracy. But with the aid of the CNN network the filters learn by themselves based on the labeled data automatically and then these learnt filters now consist of large number of parameters for the processing and designing as compared to that of the filters designed by the humans. So, with the help of these parameters it helps in classifying the training data set as they resemble the similar patterns of the object in a particular class.
Basically, the traditional methods of computer vision are divided into two categories namely the first as feature extraction and the second part as the feature classification. In the traditional method of computer vision the feature extraction is a very complex as well as computationally intensive process as it varies for different case and hence can't be generalized so a huge amount of information is required in order to perform the feature extraction step. Moving on to the second part i.e., feature classification which very much is dependent on the feature extraction step which is also complicated and complex step. Considering all these factors, the CNN model takes care of the feature extraction and the classification by learning all the features on its own and extracting them by own from the input. It does not require any complex pre-processing operation to be performed.



Also, from the example of image classification, the CNN architecture is implemented for the input data set of images and then the layers of the neural network are applied to extract the useful features from the input image by learning the filters on their own with the help of weights and biases without any demand for the prior knowledge, thereby generalizing the system. The CNN architecture is way better than the traditional methods of the computer vision techniques in the sense that it results in high accuracy, less memory usage, shift and distortion invariant and simple training.

The comparison of different machine learning techniques can be summarized in tabular form as follows:



Classification method	Advantages	Disadvantages
Artificial Neural Network	• It is a non-parametric classifier. • It is a universal functional approximator with arbitrary accuracy. • It is a data driven self-adaptive technique • Efficiently handles noisy inputs • Computation rate is high	• It is semantically poor. •The training of ANN is time consuming. • Problem of over fitting. • Difficult in choosing the type network architecture.
Decision tree	• Can handle nonparametric training data • Does not required an extensive design and training. • Provides hierarchical associations between input variables to forecast class membership and provides a set of rules n are easy to interpret. • Simple and computational efficiency is good.	•The usage of hyperplane decision boundaries parallel to the feature axes may restrict their use in which classes are clearly distinguishable. •Becomes complex calculation when various values are undecided and/or when various outcomes are correlated
Support Vector Machine	• It gains flexibility in the choice of the form of the threshold. • Contains a nonlinear transformation. • It provides a good generalization capability. • The problem of over fitting is eliminated. • Reduction in computational complexity. • Simple to manage decision rule complexity and error frequency	•Result transparency is low. •Training is time consuming. • Structure of algorithm is difficult to understand •Determination of optimal parameters is not easy when there is nonlinearly separable training data.
Fuzzy Measure	• Efficiently handles uncertainty. • Properties are described by identifying various stochastic relationships.	• Without prior knowledge output is not good • Precise solutions depend upon direction of decision
Convolutional Neural Network	• Has best-in-class performance on problems that significantly outperforms other solutions in multiple domains. This includes speech, language, vision, etc. This isn’t by a little bit, but by a significant amount.	• Requires a large amount of data — if you only have thousands of examples, deep learning is unlikely to outperform other approaches. • Is extremely computationally expensive to train. The most complex models take weeks to train

	• Reduces the need for feature engineering, one of the most time-consuming parts of machine learning practice. • It is an architecture that can be adapted to new problems relatively easily • CNN has a very layered structure where each layer considers e very nitty gritty of an image to be classified with great details and accuracy which is not done in any of the above mentioned methods. • Classification accuracy is the highest in this method.	using hundreds of machines equipped with expensive GPUs. • Do not have much in the way of strong theoretical foundation. This leads to the next disadvantage. • Determining the topology/flavor/training method/hyperparameters for deep learning is a black art with no theory to guide you. • What is learned is not easy to comprehend. Other classifiers (e.g. decision trees, logistic regression etc) make it much easier to understand what's going on.

Answer 4:

Loss Function:

Loss function can be defined as a function which is used to measure the performance of the convolutional neural network. It is basically used for the optimization process. Loss function is also known as the cost function which basically calculates the difference between desired network output and the expected network output. The lesser is the difference between the desired network output and the expected network output the better is the performance of the neural network. So, the main goal is to basically minimize the loss or the cost function for the better performance of the convolutional neural network. Here we use Cross Entropy type of loss function to determine the performance of the convolutional neural network which is given by the following expression:

$$Cross\ Entropy = -\sum_i y_i' \log(y_i)$$

Where, y' is the ground truth label and the y is the classifier prediction.
In order to minimize the cross-entropy we can adopt various optimization techniques such as stochastic gradient descent, sequential gradient descent and batch gradient descent. The parameters of the model get updated based on the loss function calculation on the training data via either back propagation technique or the chain rule. The selection of weight is done in order to minimize the cost/loss function.

Back Propagation:

As we know that the convolutional neural network performs both the tasks such as feature extraction as well as the feature classification together rather than the traditional image classification technique. Meaning that it is basically a self-learning procedure of feature extraction as well as image classification. So, technically it takes less time to achieve the

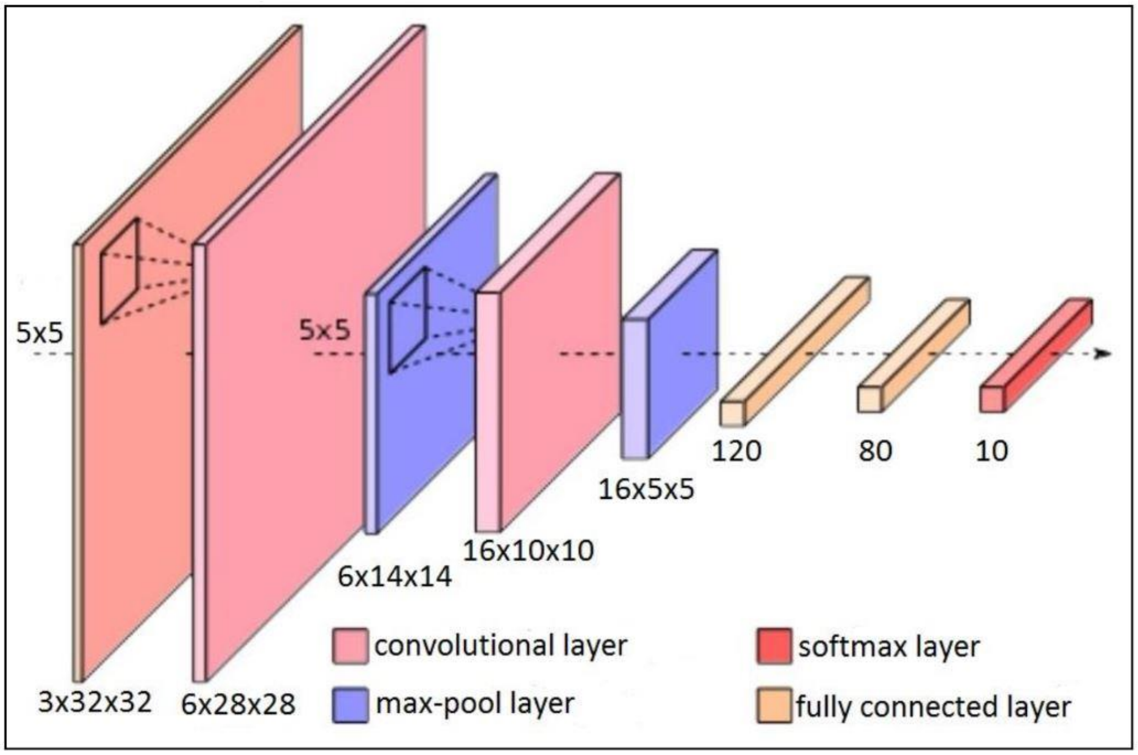
final output. The main role behind the self-learning process is the Back-Propagation technique.
Back-propagation is one of the most important way to minimize the cost function by adjusting the parameters to improve the performance of the neural network. As we know there are various non-convex optimization techniques but here, we chose the gradient descent model because of its simplicity and the robustness.
The Back-Propagation method is very commonly used in the ANN along-with the gradient descent as an optimization method. The back-propagation method is a two-stage algorithm namely the propagation and the update of the of the weights. Initially, when an input image is introduced in the network, it is carried in the forward direction in the neural network model, through one after the other layer till it reaches the last i.e., the output layer. Once it reaches the output layer then it is compared with the desired output with the aid of the loss function and then the error value is generated for each neuron of the output layer. These error values are then propagated backwards (negative gradient descent) from the output layer to the input layer direction till each neuron is associated with the error value which approximately represents its contribution to the actual final output. Basically, during the back propagation, the gradient descent for every weight is computed by basic multiplication of the input gradient with the locally gradient and then carried to the neurons till it reaches input in the backward direction fashion. With the weights getting updated in the backward direction the cost function is re-calculated in the forward direction while the GD of the cost function is moved in the backward direction as the weights gets updated.
Back-propagation method basically uses these error values in order to compute the gradient of the cost function with respect to the weights of the network. The second stage corresponds to the optimization in which the gradient calculated in the above step is fed to the optimizer in turn to update the weights and thereby minimizing the loss function. The crucial part of this operation is that as the model gets trained the neurons of the middle layers rearrange themselves such that the neurons start learning the various features of the an input dataset, so from then whenever an test input pattern is fed into the system combined with noise and the distortions the neuron of the intermediate layers will generate an output if the test input fed contains the pattern or feature that comprises of the pattern or feature such that the neurons have learned to identify them by their own during their learning phase. Back-propagation method generally requires the pre-determined output for each particular kind of input in order to compute the cost function, therefore it is known as the supervised learning process. The main aim of the back-propagation technique is to implement the way of training a multi layered convolutional neural network in such a way that the internal parameters are learnt by the layers by their self without the use of any kind of external information so that the mapping of any random input data to the output can be done.
Basically, this back-propagation step continues till the gradient descent of the loss function gets converged. The computational complexity of the BP algorithm is very low and hence efficient. It also uses less amount of memory for storing if applied to any kind of neural network due to the implementation of the chain rule.

Mathematically, the backpropagation is computed via partial derivative or may be known as the gradient of loss function w.r.t the weight vector in the neural network.

(b) CIFAR-10 Classification:

1. ABSTRACT & MOTIVATION

LeNet-5 is basically a feed forward convolutional neural network in which the artificial neurons respond to the nearby cells thereby performing well in the large-scale image processing. Convolutional Neural Network is technically a self-learning process which means that it extracts the features and then will classify them automatically, which in turn results in less classification time but computationally intensive.

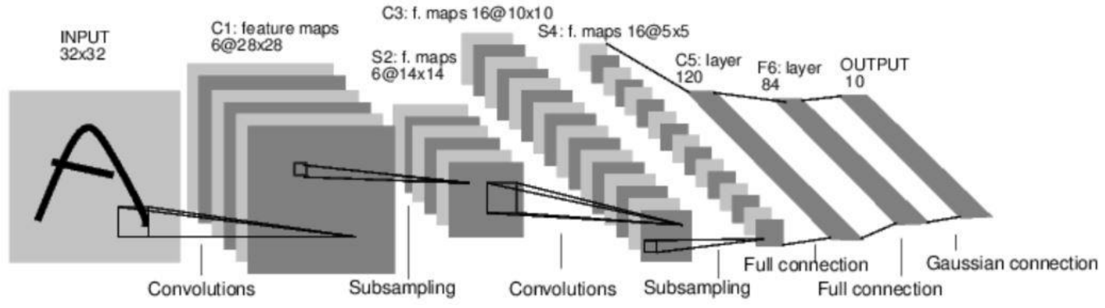


Since CNN can be computationally intensive, so for big dataset and heavy tasks, in practice all such kind of machine learning activities are carried out using high performance GPUs and multi core processors. But here as we have relatively smaller dataset of CIFAR-10 which contains 50,000 training data images and 10,000 testing data images where each train as well as test images are of 32x32 dimension, so I can carry out this machine learning task on my system only. So, in this problem I am required to train a simple convolutional neural network called **LeNet-5** invented by Yann LeCun the director of Facebook AI research which was basically designed for handwritten and machine printed character recognition. Since we know that the LeNet-5 model of CNN architecture was designed mainly for character recognition and was not meant for image recognition or image classification so it is obviously expected to have lower accuracy to be obtained at last using this architecture for image recognition or image classification as compared to the accuracy of the LeNet-5 model used for character recognition.

2. APPROACH & PROCEDURE

LeNet-5 architecture has basically two spatial convolutional layers, two max pooling layers along-with the two fully connected hidden layers where each layer consists of various number of parameters and one input and one output layer. It has 3 connection methods as well.

The functionality of the layers for the CIFAR-10 dataset can be explained in detail as follows:



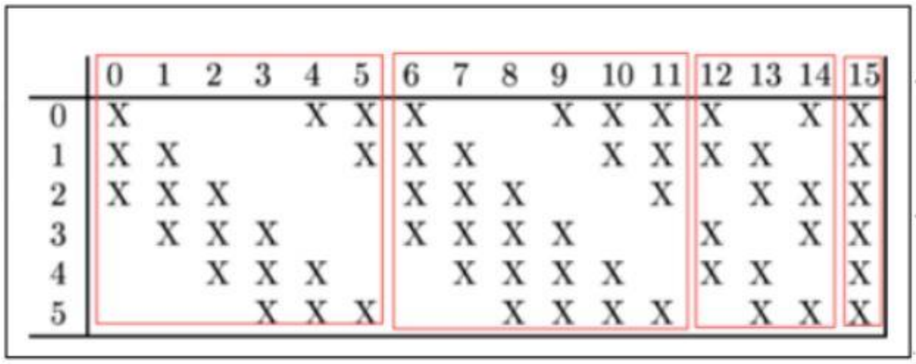
- **Input Layer:** This is the first and the foremost layer of the LeNet-5 neural network which consists of the input images. The size of this layer is dependent on the individual size of the images of the training as well as the testing dataset. So, here as we are dealing with the CIFAR-10 image dataset which consists of 60,000 of the RGB images where the dimension of each image is 32x32, out of which the 50,000 images belong to the training dataset which are labelled and the remaining 10,000 images belong to the testing dataset. The depth of this dataset is 3 as the images are RGB so the spatial arrangement of the neurons has a depth of 3.

- **Spatial Convolutional Layer 1:** In this layer of the model which is just after the input layer acts as the initial layer of the network. Here, the input images are convoluted with the filters using the 5x5 kernels to obtain the feature maps, where each feature map corresponds to a particular characteristic feature. As the input to this layer has high dimensional data, in this case it is 32x32x3x60,000 = 1,84,320,000. With such a large amount of neural data as input it results in very huge amount of computations thereby leading to 4,821,600 weights which needs to be learnt by the first layer itself. Due to such large amount of data it may result in poor output results. So, in order to overcome this scenario, we adopt a better approach in which the local region of an input image from the input dataset is chosen which is then connected to each neuron in the next layer. The advantage of this approach is the reduction in the receptive field of each neuron for the next layers. Here, we choose our receptive field i.e., the kernel size to be 5x5 which means that the output of each neuron will see the input to be 5x5. For, the LeNet-5 architecture we choose our receptive field to be 5x5 with no zero padding and the stride parameter for this layer as 1. Each of the input image is convolved with 6 filters of this convolutional layer in order to obtain 6 feature maps corresponding to each input image. Thus, the 32x32 dimension of an input image gets transformed to the 28x28 image with 6 feature maps for each input image where each layer represents the convoluted output for each particular filter. The calculation for the amount of training parameters for this layer is 26x6 = 156 and (5x5 + 1) x28x28x6 = 122304 connections.

- **Spatial Max Pooling Layer 1:** In this layer the output of the first spatial convolutional layer is fed as the input to this Max pooling Layer 1. The main purpose of this layer is basically to enhance the receptive field with respect to the input layer thereby reducing the number of weights for the succeeding layers. Basically, the principle of local correlation used i.e., the sub-sampling (down-sampling) the image will reduce the amount of data processing along-with retaining the useful data information required for the processing for the next steps. So, here the kernel size is considered as 2x2, which means that the output we get from this Max-pooling layer 1 will be 6 feature maps each of dimension 14x14, thereby having 5x14x14x6 = 5880 connections.

- **ReLU Non-Linear Activation 1:** The main purpose of using ReLU Non-Linear activation function rather than using the other non-linear activation function is because through various experimental results it has been found the ReLU non-linear activation function tends to converge quickly as compared to other kinds of non-linear activation functions. Basically, in this layer the image size is not changed, and the weights are also not learned in this layer. The purpose of this layer is to apply activation elementwise such that it is threshold to zero. As we know that nothing is linear in this world so in order to introduce some non-linearity in the network model, we use this non-linear activation function in order to improve the performance accuracy of the neural network. Thus, the representation of the ReLU function is given as **output = max (0, input).**

- **Spatial Convolutional Layer 2:** The convolution operation of this layer is similar to the first spatial convolutional layer. The input to this layer is the output of the first max pooling layer where the dimension is 14x14x6 and the convolution is done with the receptive field of 5x5. The only difference of this layer is that the output of this layer gives the 16 layered features maps on the basis of 16 filters which are generated using the asymmetric combination connection method in order to extract the combination of various features whose logic given below.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

So basically, this layer will have (5x5x3 + 1) x6 + (5x5x4 + 1) x6 + (5x5x4 + 1) x3 + (5x5x6 + 1) x1 = 1516 training parameters with the total of 1516x10x10 = 151600 connections.

- **Spatial Max Pooling Layer 2:** The input fed to this layer is the output of the spatial convolutional layer 2. The operation performed in this layer is also analogous to that of the spatial max pooling layer 1, using 2x2 window size. So, as a result the output of this layer will obtain 16x5x5 dimensional images. No weights are there in this layer. This layer has 5x5x5x16 = 2000 connections.

- **ReLU Non-linear Activation 2:** The performance of this layer is same as that of the ReLU non-linear activation 1. The only difference is that the dimension of the input image is changed from 32x32 = 1024 to 20x20 = 400 neurons. Till this step the feature reduction as well as dimensionality reduction is achieved and from here the images are passed through the fully connected layers.

- **Fully Connected Hidden Layer 1:** So, as per the architecture given in the problem statement the first fully connected layer consists of 120 neurons which is linked to the 400 neurons of the previous layer. So, basically as there are 16 5x5 dimensional feature maps in the previous layer which is of the same size as that of the convolutional kernel, the dimension of the output obtained from this layer is 1x1. The total number of learning weights are given as (5x5x16 + 1) x 120 = 48120. The unique thing about this layer is that it has maximum receptive field, which means that each and every neuron has the complete information about the input image.

- **Fully Connected Hidden Layer 2:** This principle of this layer is same as that of the above fully connected layer 1 with the difference here that the input to this layer is 1 dimensional which means that the 120 neurons from the previous layer will serve as the input to this layer. These 120 neurons need to be connected in a fashion such that they are connected to the 84 nodes in this layer. 84 nodes correspond to 7x12 bitmap and the number of training parameters and the connections are given as (120 + 1) x 84 = 10164.

- **Output Layer:** This the last layer of the LeNet-5 neural network model which consists of 10 neurons because of the 10 classes of the CIFAR-10 dataset which is in turn connected to the 80 neurons of the previous layer. The number of learning weights in this layer is given as 84x10 = 840. The softmax function is used in this last layer in order to give the probabilities for the output labels. The softmax function is applied to each neuron in order to normalize the output which helps in determining the output labels easily. The log softmax function is represented as $s(x) = \log\left(\frac{e^x}{\sum_{i=1}^{N} e^i}\right)$.

  This step also reduces the cross- entropy by passing the output of each neuron to the above given function.

  Hence, here is the complete explanation of the entire LeNet-5 convolutional neural network which has 60,840 training parameters and 340,908 connections.

3. EXPERIMENTAL RESULTS

Number of Epochs = 100
Batch size = 128

Sr. No.	**Optimizer**	**Dropout**	**Learning rate**	**Kernel initializer**	**Bias initializer**	**Train loss**	**Train accuracy**	**Test loss**	**Test accuracy**
1.	RMS prop	0.5	0.001	Glorot normal	Random Normal	0.9052	0.6778	1.089776	0.623700
2.					Random Uniform	0.8842	0.6834	1.068800	0.633200
3.				Glorot Uniform	Random Normal	0.8860	0.6820	1.012571	0.645900
4.					Random Uniform	0.8734	0.6850	1.062861	0.634000
5.			0.01	Glorot normal	Random Normal	1.7976	0.3601	1.773513	0.363600
6.					Random Uniform	1.7443	0.3784	2.004609	0.267800
7.				Glorot Uniform	Random Normal	1.8381	0.3432	1.814103	0.367800
8.					Random Uniform	1.8077	0.3525	1.714688	0.347200
9.				Glorot normal	Random Normal	0.6117	0.7787	1.224225	0.634700
10.					Random Uniform	0.6303	0.7733	1.257270	0.623700

EE569 Digital Image Processing

11.		0.25	0.001	Glorot Uniform	Random Normal	0.6138	0.7811	1.352221	0.615400
12.					Random Uniform	0.6417	0.7685	1.337297	0.610800
13.			0.01	Glorot normal	Random Normal	1.7441	0.3897	1.864878	0.357900
14.					Random Uniform	1.6429	0.4265	1.803015	0.363100
15.				Glorot Uniform	Random Normal	1.7566	0.3833	1.606204	0.439300
16.					Random Uniform	1.6937	0.4076	1.532952	0.442500
17.	SGD	0.5	0.001	Glorot normal	Random Normal	1.1882	0.5778	1.140893	0.592500
18.					Random Uniform	1.1145	0.6044	1.079544	0.620900
19.				Glorot Uniform	Random Normal	1.1223	0.5980	1.104240	0.606100
20.					Random Uniform	1.1246	0.5985	1.075304	0.614700
21.			0.01	Glorot normal	Random Normal	0.8805	0.6838	1.058806	0.631000
22.					Random Uniform	0.8363	0.7016	1.060316	0.629500
23.				Glorot Uniform	Random Normal	0.8644	0.6933	1.056857	0.635700
24.					Random Uniform	0.8529	0.6918	1.050262	0.628900
25.		0.25	0.001	Glorot normal	Random Normal	1.0174	0.6408	1.092912	0.615900
26.					Random Uniform	1.0076	0.6436	1.069318	0.625100
27.				Glorot Uniform	Random Normal	0.9518	0.6635	1.037804	0.637700
28.					Random Uniform	1.0050	0.6464	1.064076	0.629100
29.			0.01	Glorot normal	Random Normal	0.8453	0.6965	1.082755	0.626700
30.					Random Uniform	0.5817	0.7913	1.280241	0.636200
31.				Glorot Uniform	Random Normal	0.6069	0.7806	1.281408	0.616600
32.					Random Uniform	0.5958	0.7840	1.303737	0.616500
33.					Random Normal	0.8580	0.6891	1.049529	0.633500

34.	Adam	0.5	0.001	Glorot normal	Random Uniform	0.8501	0.6940	1.076643	0.643200
35.				Glorot Uniform	Random Normal	0.8729	0.6853	1.025636	0.642300
36.					Random Uniform	0.6165	0.7899	1.0775	0.63679
37.			0.01	Glorot normal	Random Normal	1.5542	0.4442	1.513650	0.458900
38.					Random Uniform	2.3030	0.1000	2.3030	0.10000
39.				Glorot Uniform	Random Normal	0.8644	0.6933	1.056857	0.635700
40.					Random Uniform	2.3034	0.1000	2.3034	0.10000
41.		0.25	0.001	Glorot normal	Random Normal	1.0174	0.6408	1.092912	0.615900
42.					Random Uniform	0.9006	0.6862	1.5379	0.5109
43.				Glorot Uniform	Random Normal	0.9518	0.6635	1.037804	0.637700
44.					Random Uniform	0.4156	0.8620	1.2757	0.6148
45.			0.01	Glorot normal	Random Normal	0.5817	0.7913	1.280241	0.636200
46.					Random Uniform	1.1787	0.5857	1.4972	0.4911
47.				Glorot Uniform	Random Normal	0.6069	0.7806	1.281408	0.616600
48.					Random Uniform	0.8986	0.8986	1.3947	0.5770

Results:

The plots are arranged in the **sequence** as per the data tabulated in table given above.

i. **RMSprop: (decay rate = 1e-6)**

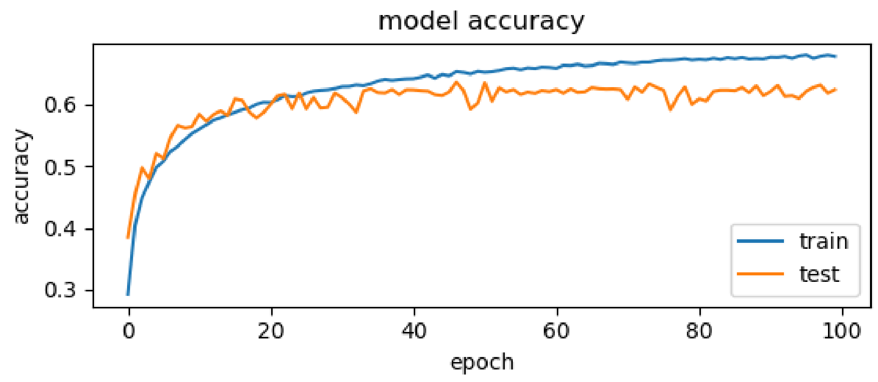

model accuracy
accuracy
0.3
0.4
0.5
0.6
0
20
40
60
80
100
epoch
train
test

Fig. 1

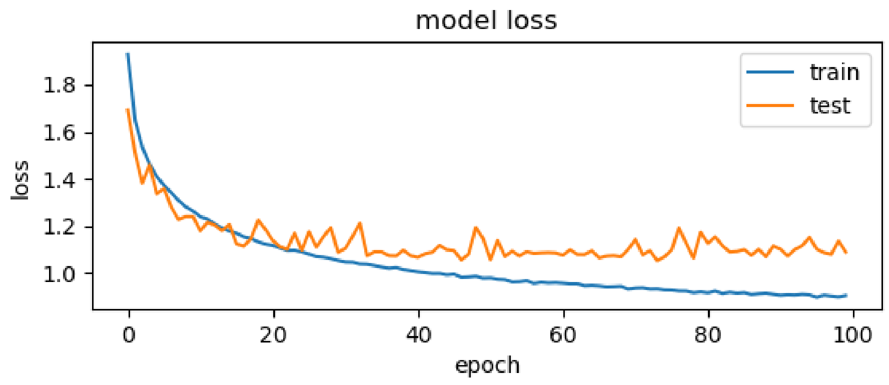

model loss
loss
1.0
1.2
1.4
1.6
1.8
0
20
40
60
80
100
epoch
train
test

Fig. 2

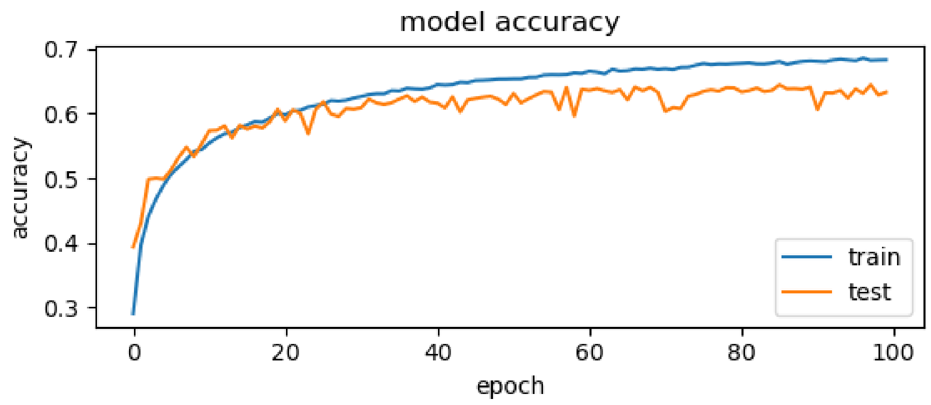

model accuracy
accuracy
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 3

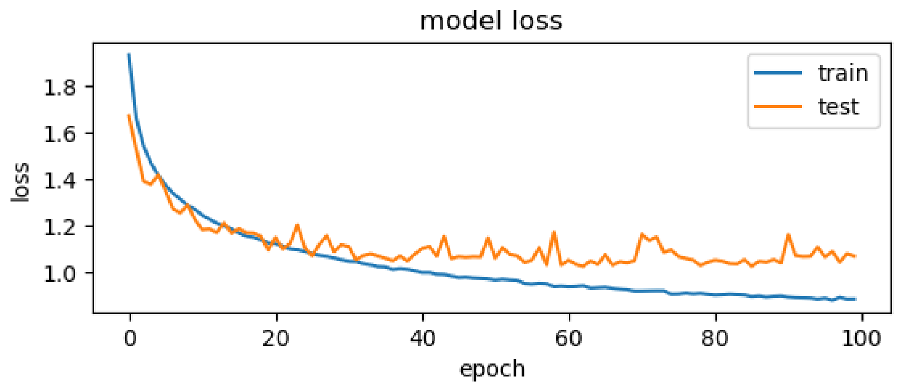

model loss
loss
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100
epoch
train
test

Fig. 4

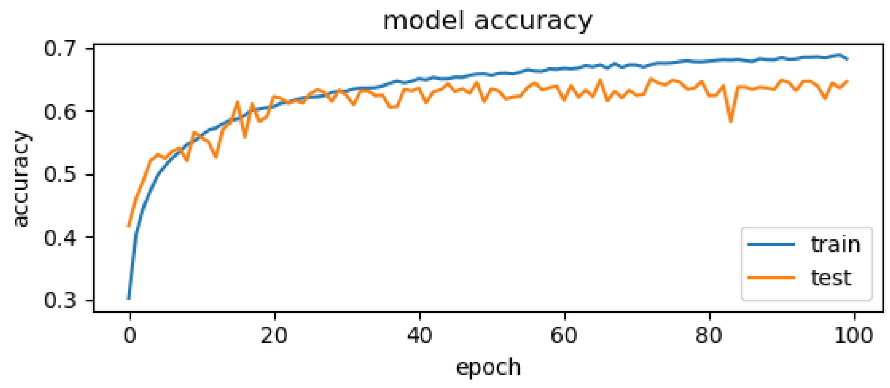

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Fig. 5

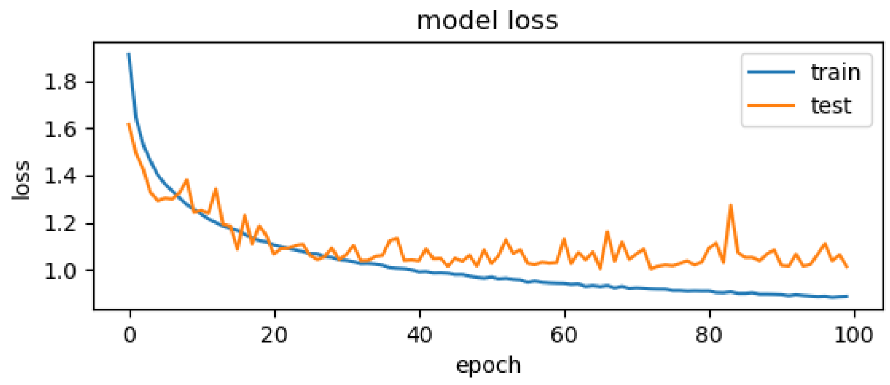

model loss
loss
epoch
train
test
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

Fig. 6

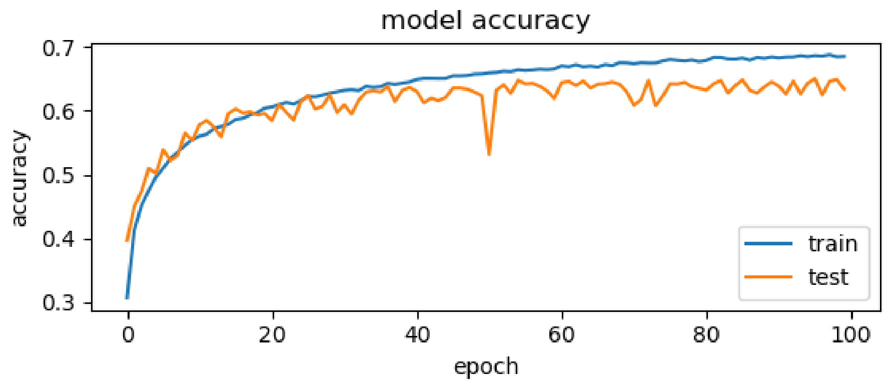

model accuracy
accuracy
0.3
0.4
0.5
0.6
0.7
epoch
0
20
40
60
80
100
train
test

Fig. 7

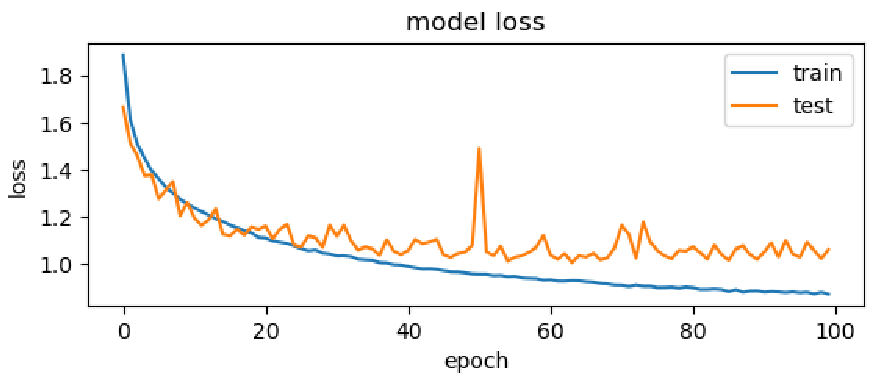

model loss
loss
1.0
1.2
1.4
1.6
1.8
epoch
0
20
40
60
80
100
train
test

Fig. 8

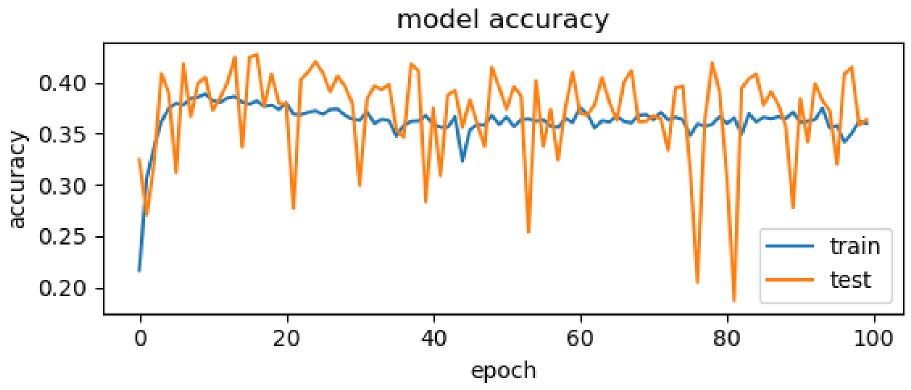

model accuracy
accuracy
0.40
0.35
0.30
0.25
0.20
0
20
40
60
80
100
epoch
train
test

Fig. 9

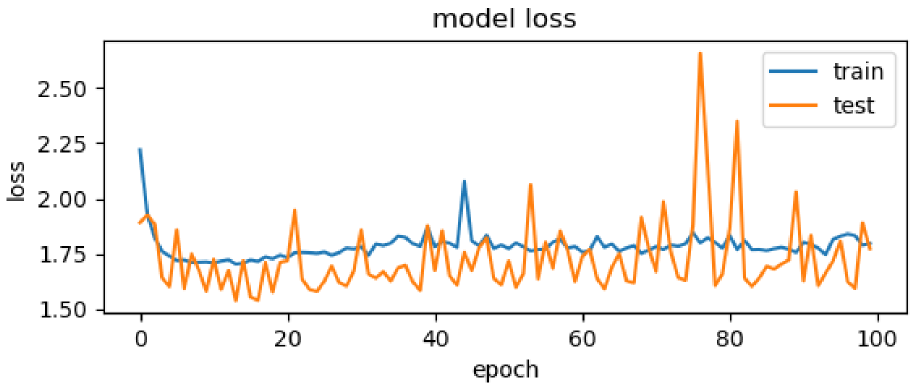

model loss
loss
2.50
2.25
2.00
1.75
1.50
0
20
40
60
80
100
epoch
train
test

Fig. 10

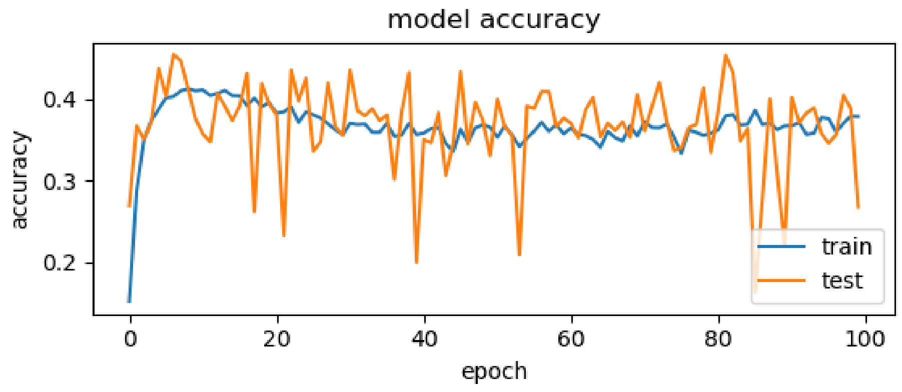

model accuracy
accuracy
0.4
0.3
0.2
0
20
40
60
80
100
epoch
train
test

Fig. 11

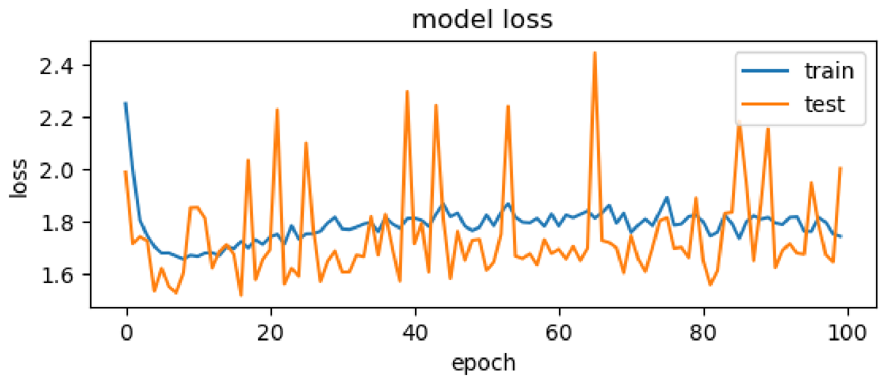

model loss
loss
2.4
2.2
2.0
1.8
1.6
0
20
40
60
80
100
epoch
train
test

Fig. 12

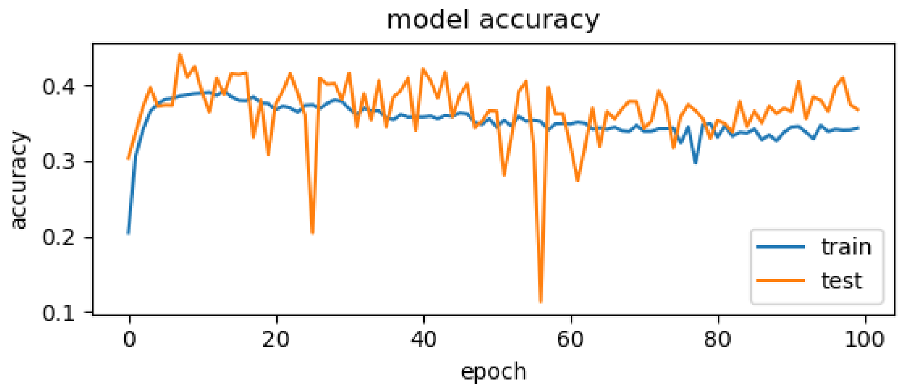

model accuracy
accuracy
0.1
0.2
0.3
0.4
epoch
0
20
40
60
80
100
train
test

Fig. 13

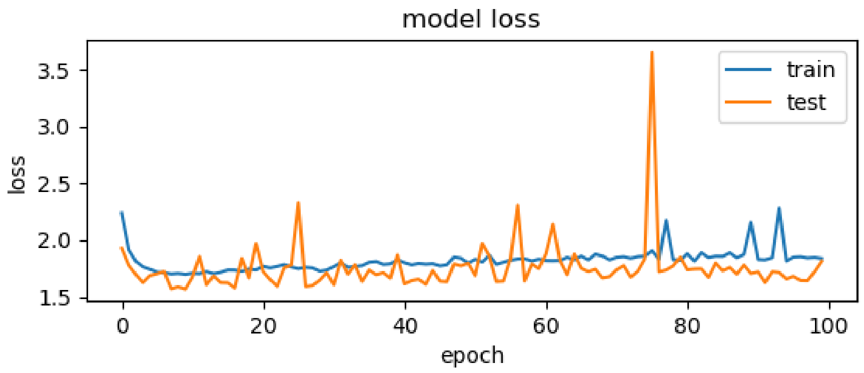

model loss
loss
1.5
2.0
2.5
3.0
3.5
epoch
0
20
40
60
80
100
train
test

Fig. 14

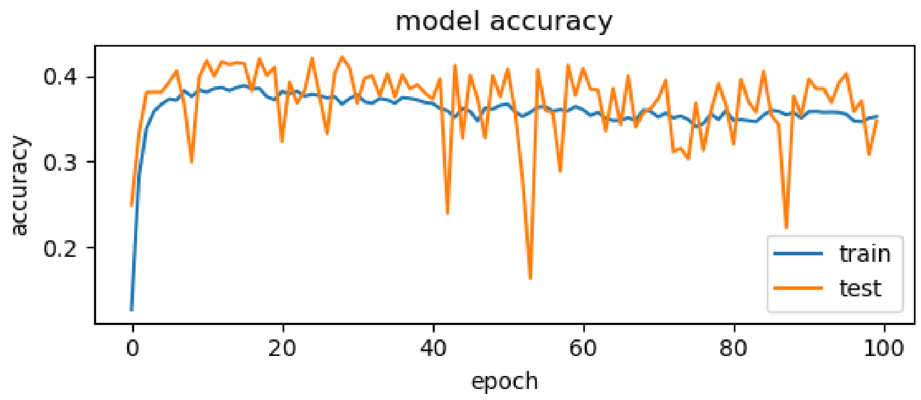

model accuracy
accuracy
0.2
0.3
0.4
epoch
0
20
40
60
80
100
train
test

Fig. 15

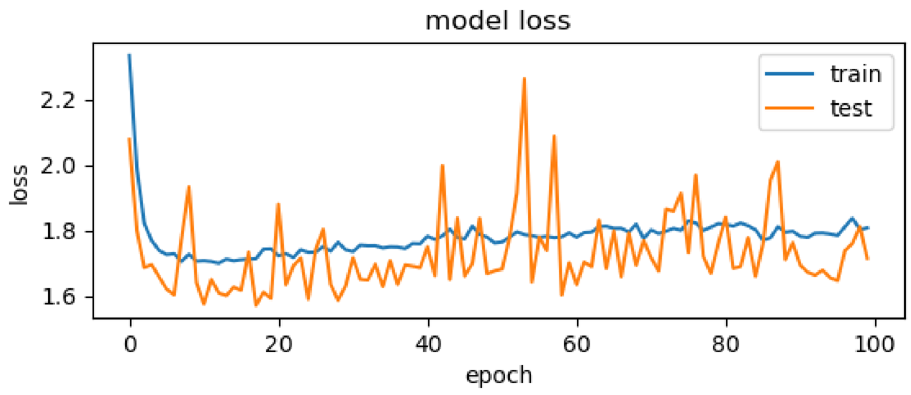

model loss
loss
1.6
1.8
2.0
2.2
epoch
0
20
40
60
80
100
train
test

Fig. 16

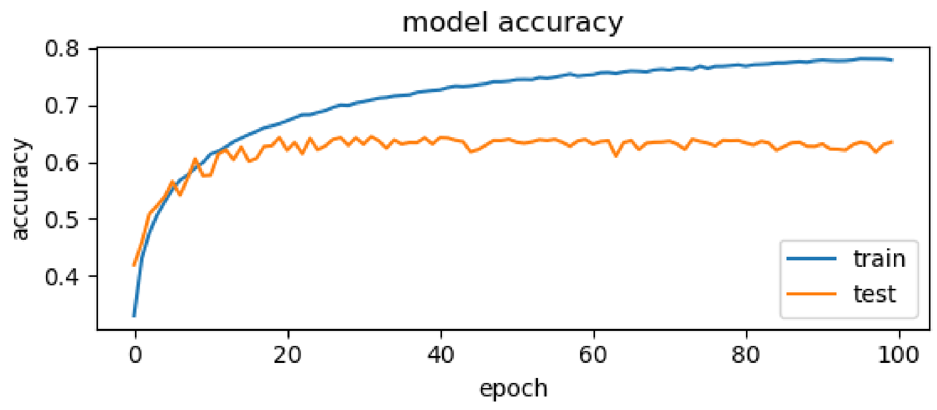

model accuracy
accuracy
epoch
train
test
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100

Fig. 17

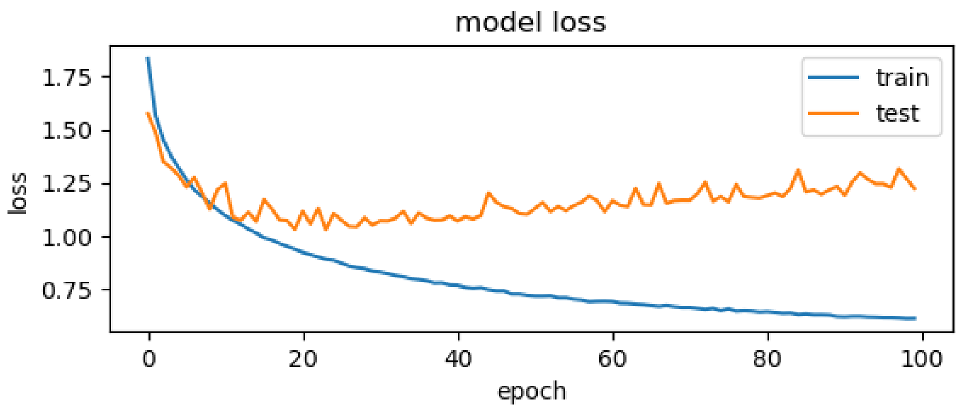

model loss
loss
epoch
train
test
1.75
1.50
1.25
1.00
0.75
0
20
40
60
80
100

Fig. 18

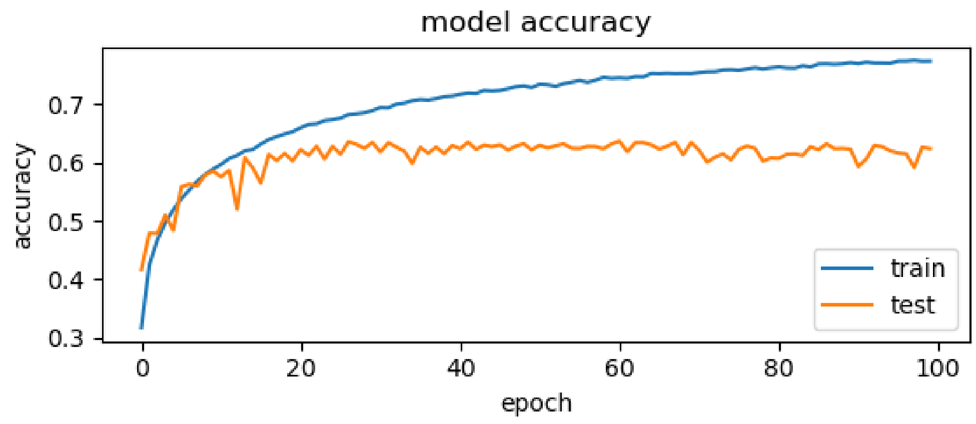

model accuracy
accuracy
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
epoch
train
test

Fig. 19

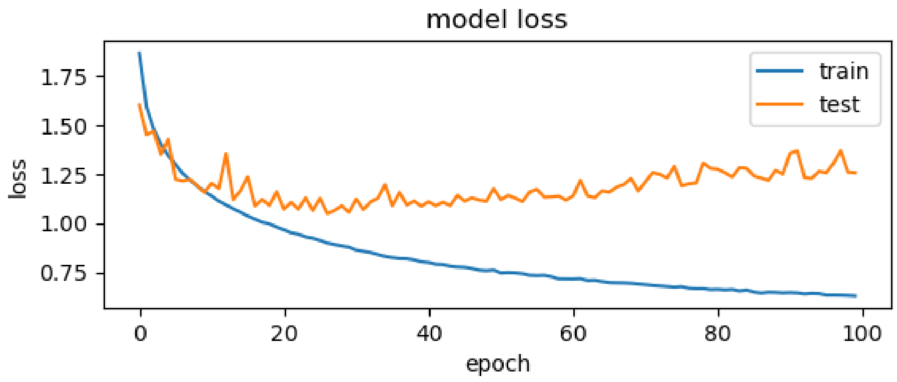

model loss
loss
0.75
1.00
1.25
1.50
1.75
0
20
40
60
80
100
epoch
train
test

Fig. 20

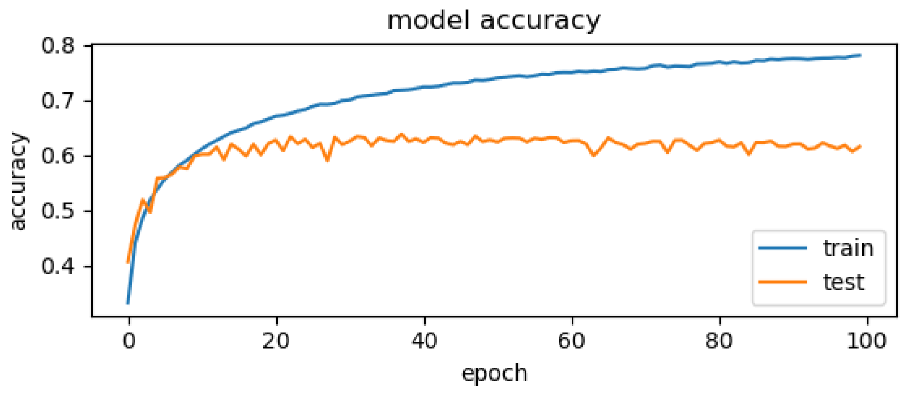

model accuracy
accuracy
epoch
train
test
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100

Fig. 21

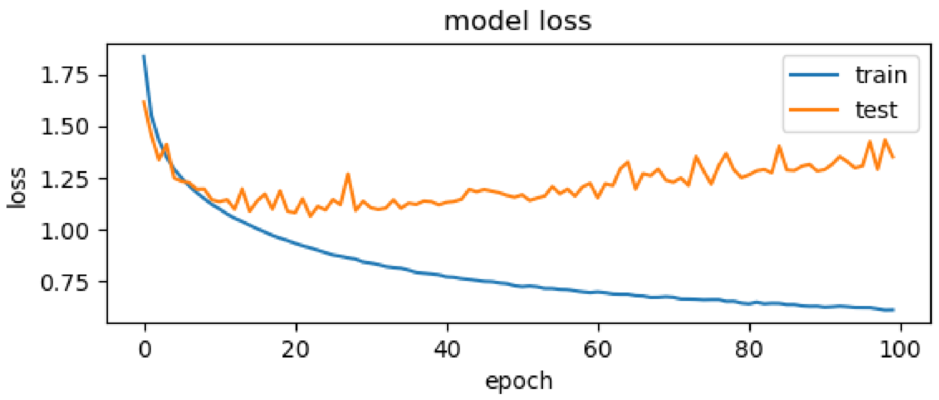

model loss
loss
epoch
train
test
1.75
1.50
1.25
1.00
0.75
0
20
40
60
80
100

Fig. 22

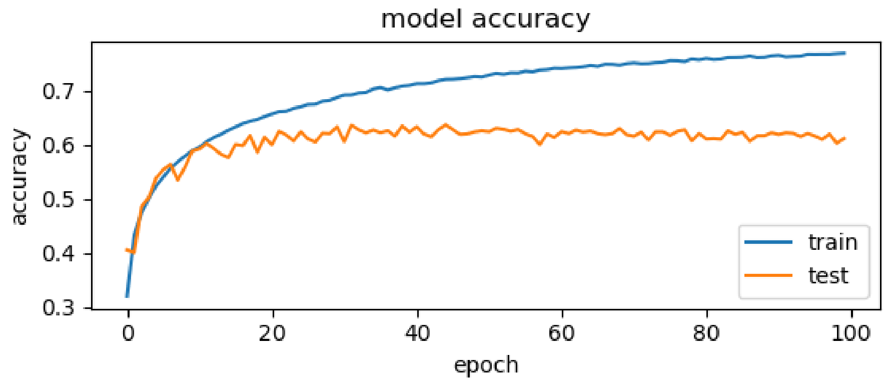

model accuracy
accuracy
epoch
train
test
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100

Fig. 23

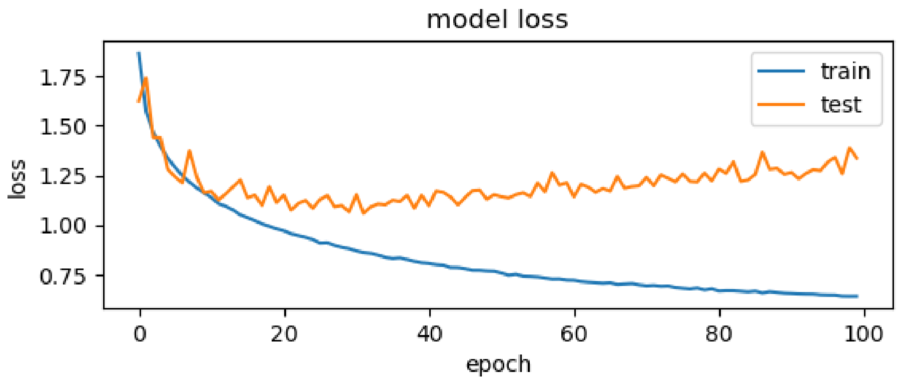

model loss
loss
epoch
train
test
0.75
1.00
1.25
1.50
1.75
0
20
40
60
80
100

Fig. 24

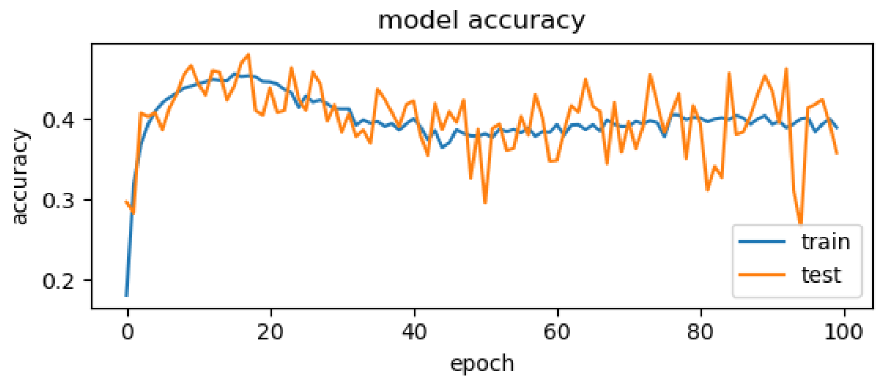

model accuracy
accuracy
0.2
0.3
0.4
epoch
0
20
40
60
80
100
train
test

Fig. 25

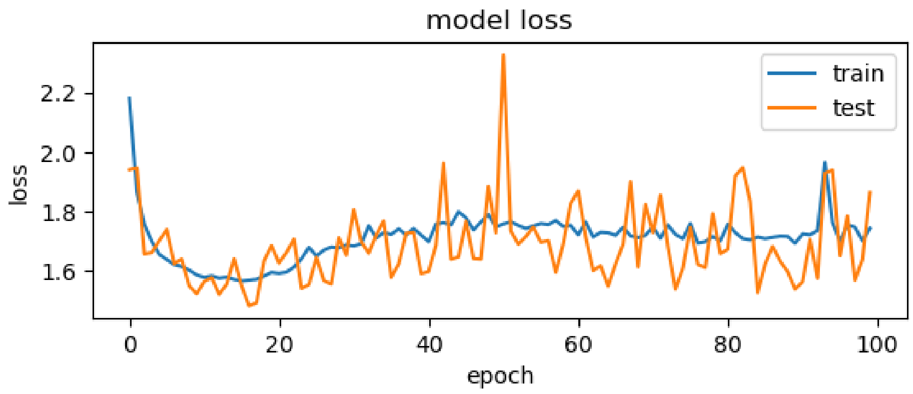

model loss
loss
1.6
1.8
2.0
2.2
epoch
0
20
40
60
80
100
train
test

Fig. 26

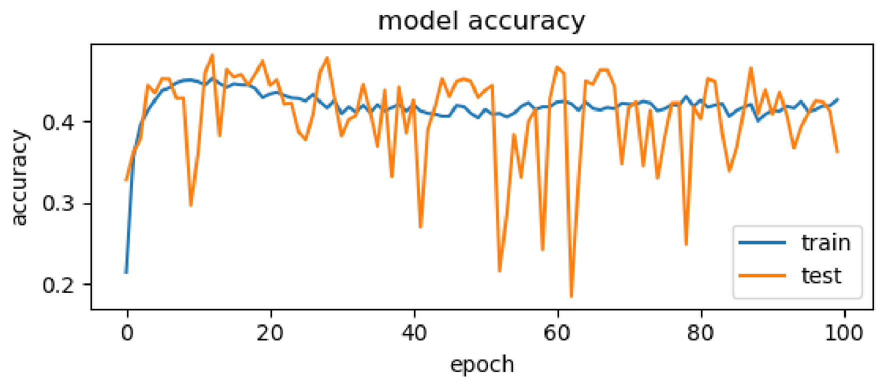

model accuracy
accuracy
0.2
0.3
0.4
0
20
40
60
80
100
epoch
train
test

Fig. 27

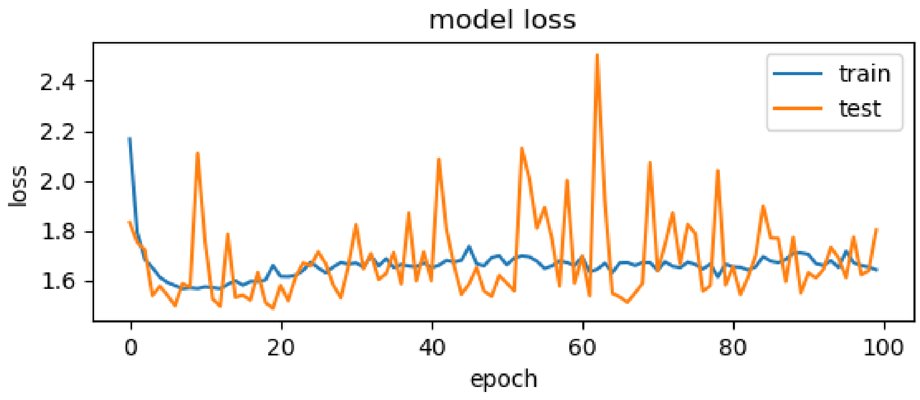

model loss
loss
1.6
1.8
2.0
2.2
2.4
0
20
40
60
80
100
epoch
train
test

Fig.28

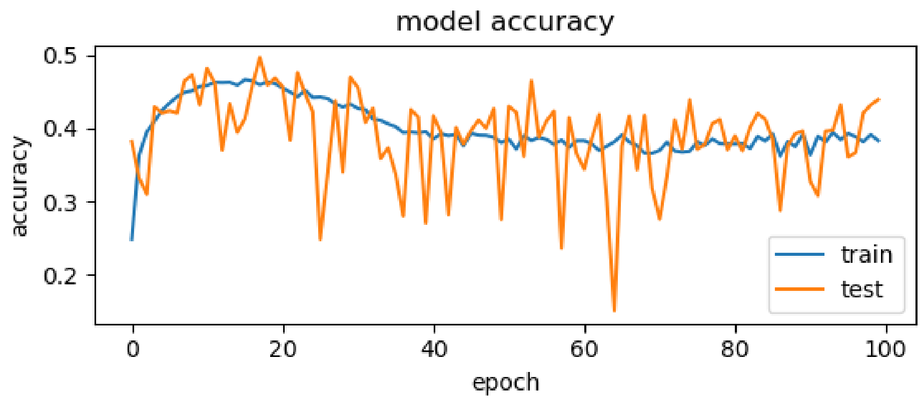

model accuracy
accuracy
0.5
0.4
0.3
0.2
0
20
40
60
80
100
epoch
train
test

Fig. 29

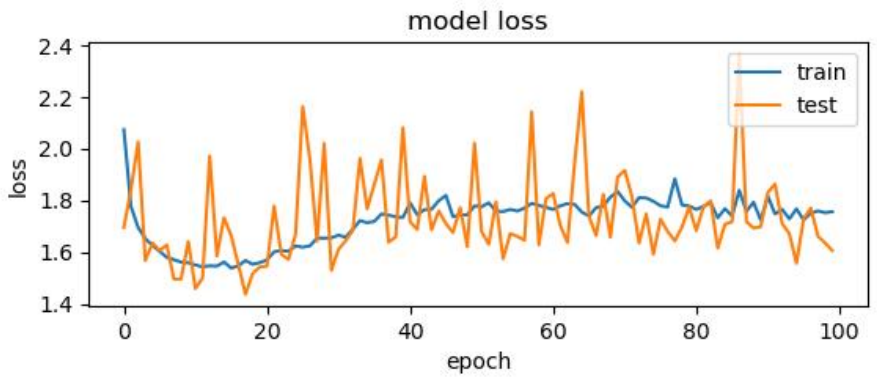

model loss
loss
2.4
2.2
2.0
1.8
1.6
1.4
0
20
40
60
80
100
epoch
train
test

Fig. 30

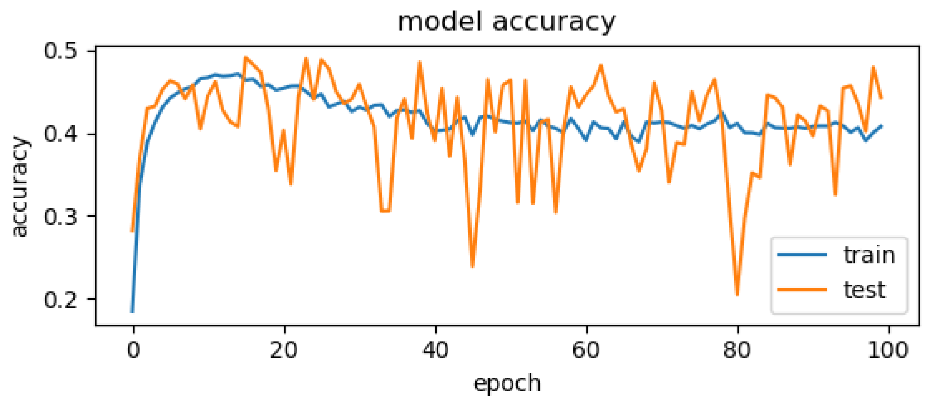

model accuracy
accuracy
epoch
train
test
0.2
0.3
0.4
0.5
0
20
40
60
80
100

Fig. 31

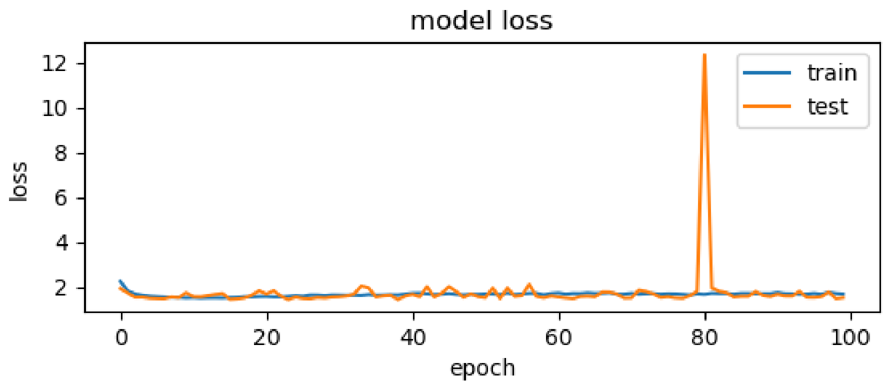

model loss
loss
epoch
train
test
2
4
6
8
10
12
0
20
40
60
80
100

Fig. 32

ii. SGD: (momentum = 0.9)

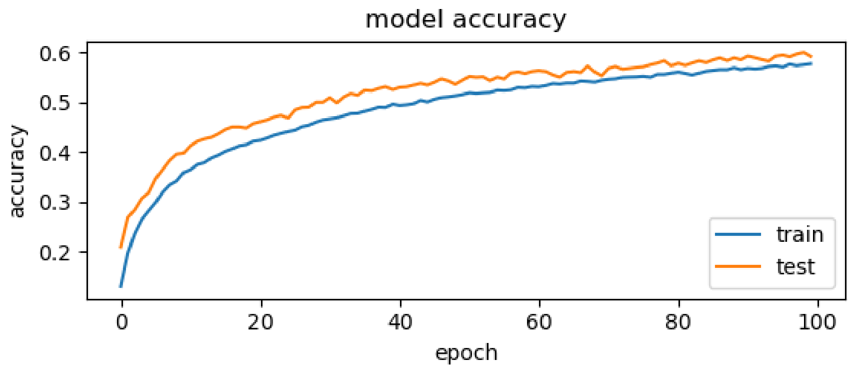


Fig. 33

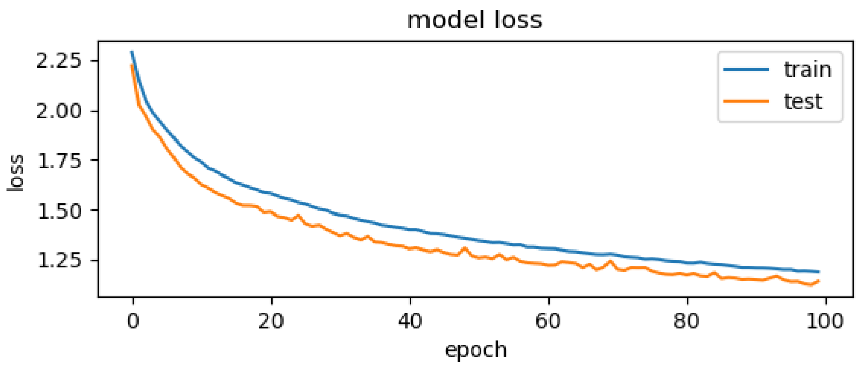


Fig. 34

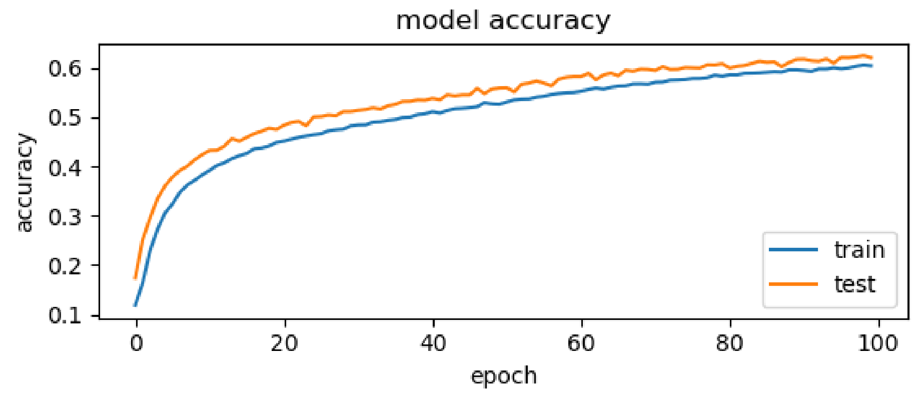

model accuracy
accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
epoch
train
test

Fig. 35

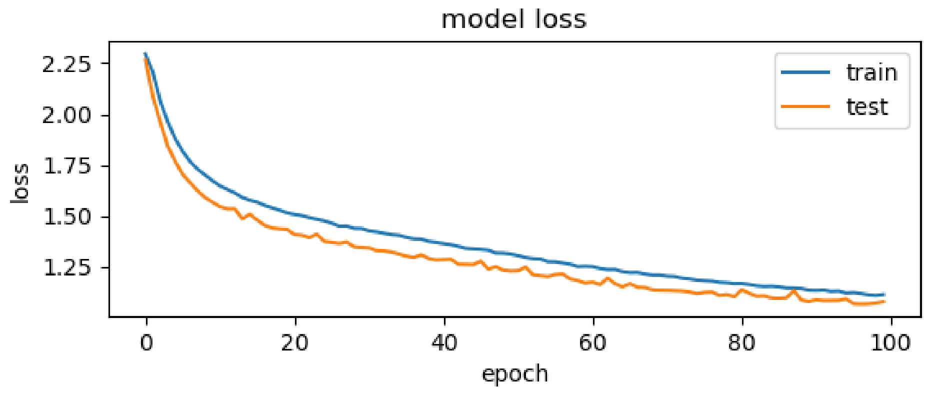
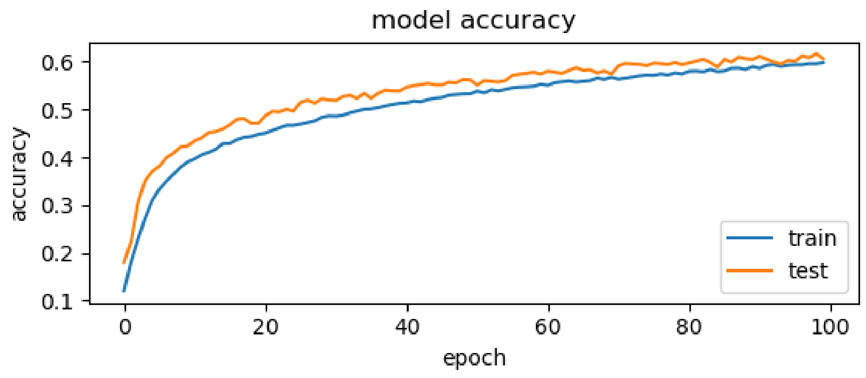
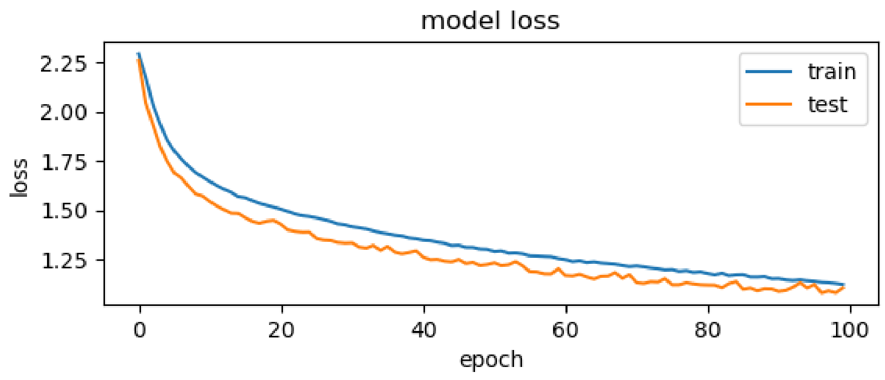

model loss
loss
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100
epoch
train
test

Fig. 36


model accuracy
accuracy
epoch
train
test
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

<u>Fig. 37</u>


model loss
loss
epoch
train
test
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100

<u>Fig. 38</u>

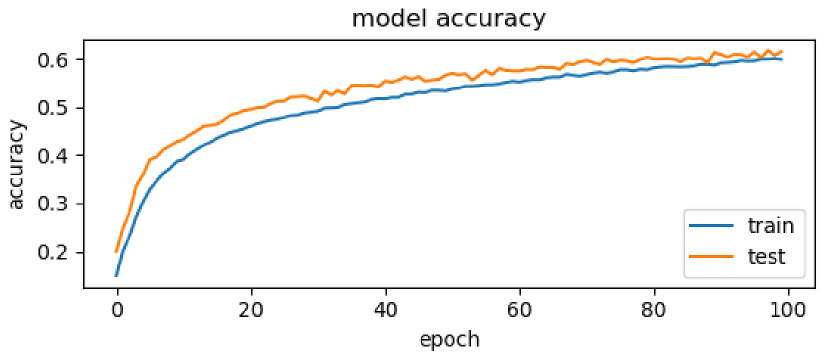

model accuracy
accuracy
0.2
0.3
0.4
0.5
0.6
epoch
0
20
40
60
80
100
train
test

Fig. 39

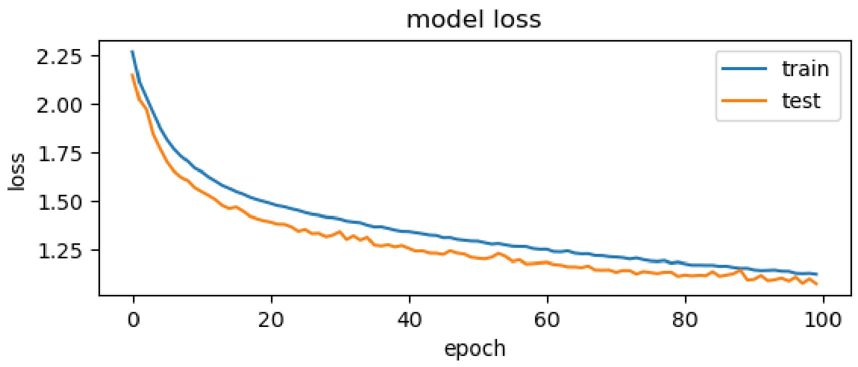

model loss
loss
1.25
1.50
1.75
2.00
2.25
epoch
0
20
40
60
80
100
train
test

Fig. 40

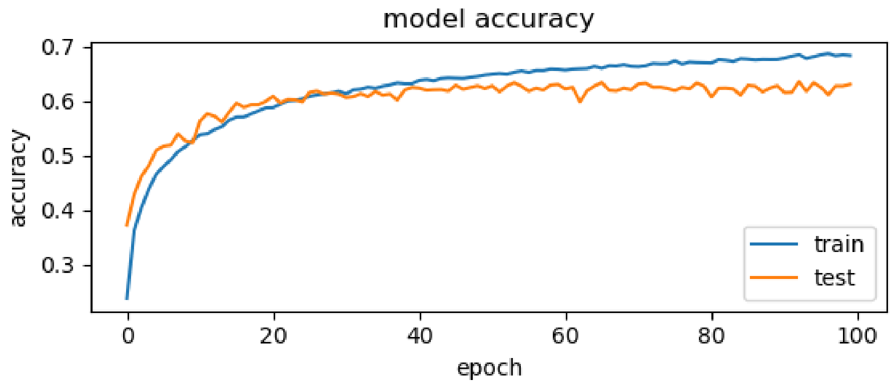

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Fig. 41

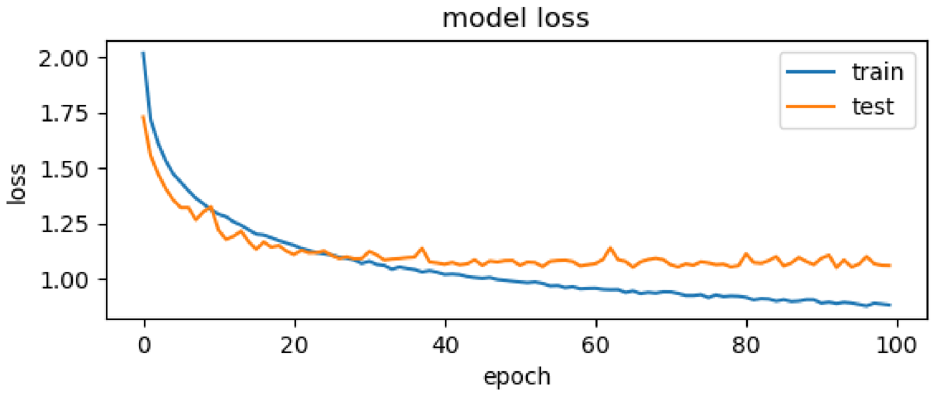

model loss
loss
epoch
train
test
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100

Fig. 42

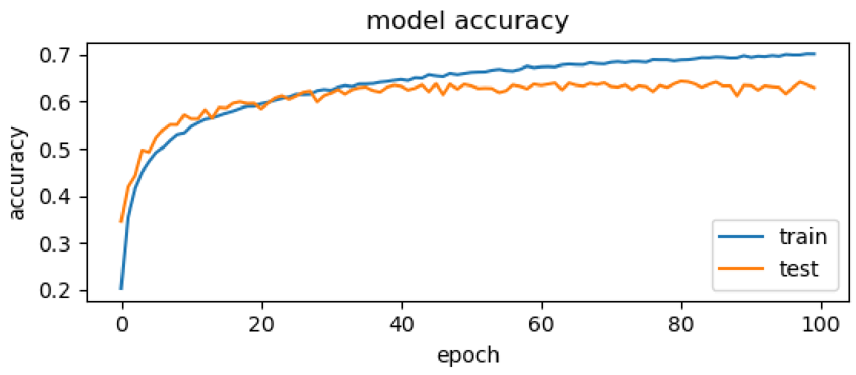

model accuracy
accuracy
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
epoch
train
test

Fig. 43

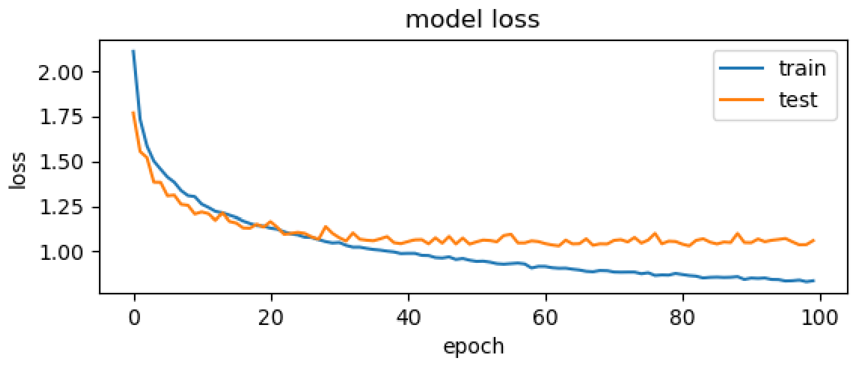

model loss
loss
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100
epoch
train
test

Fig. 44

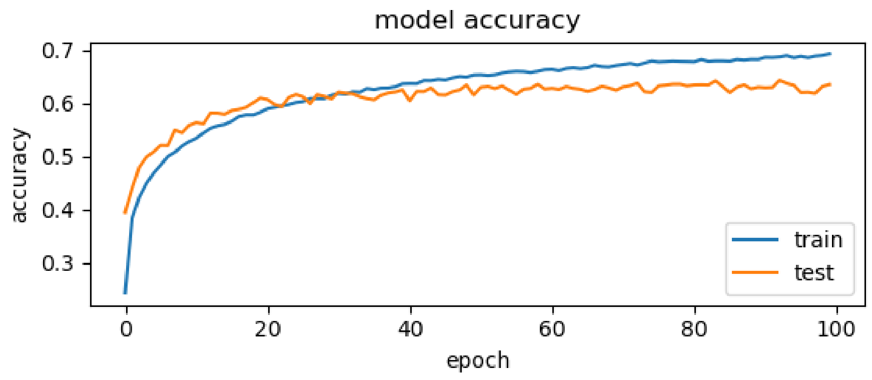

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Fig. 45

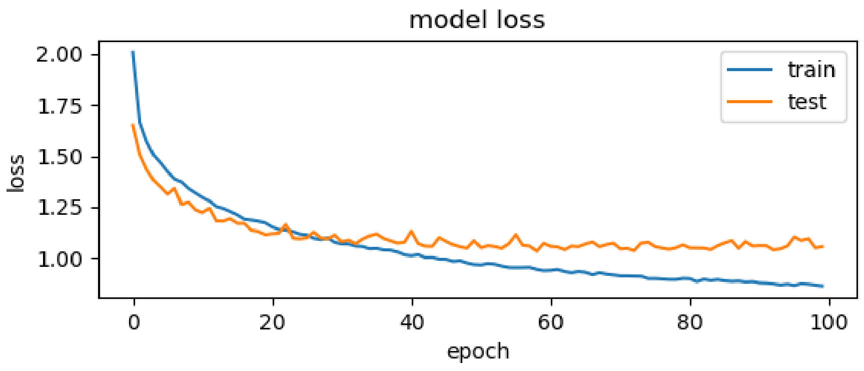

model loss
loss
epoch
train
test
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100

Fig. 46

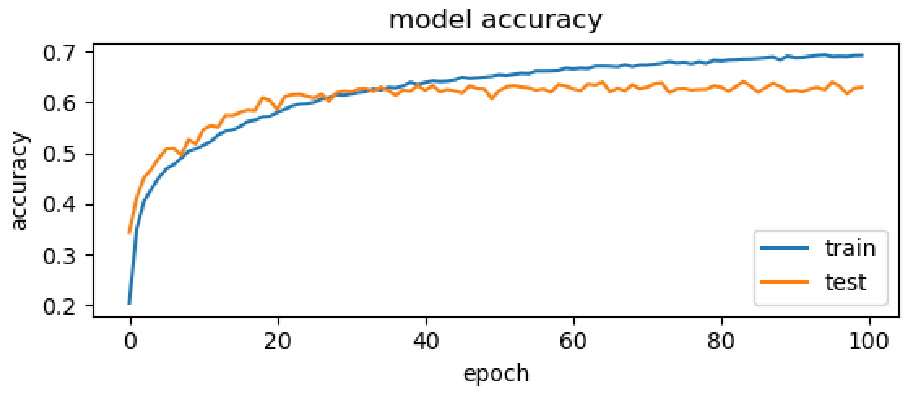

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100

Fig. 47

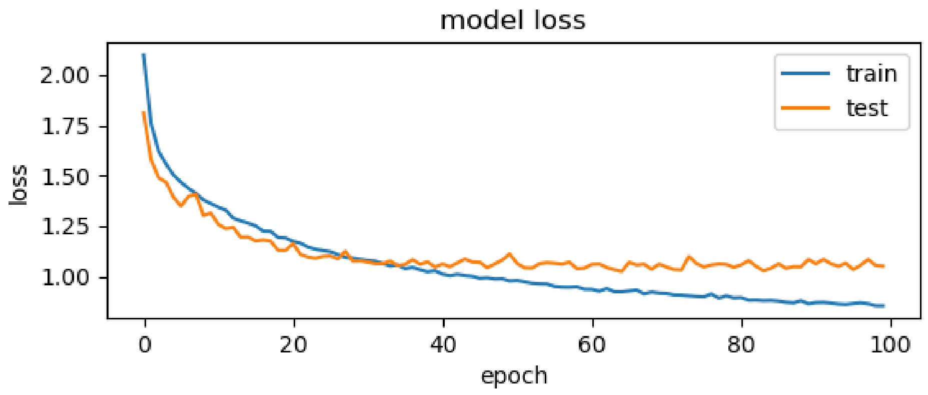

model loss
loss
epoch
train
test
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100

Fig. 48

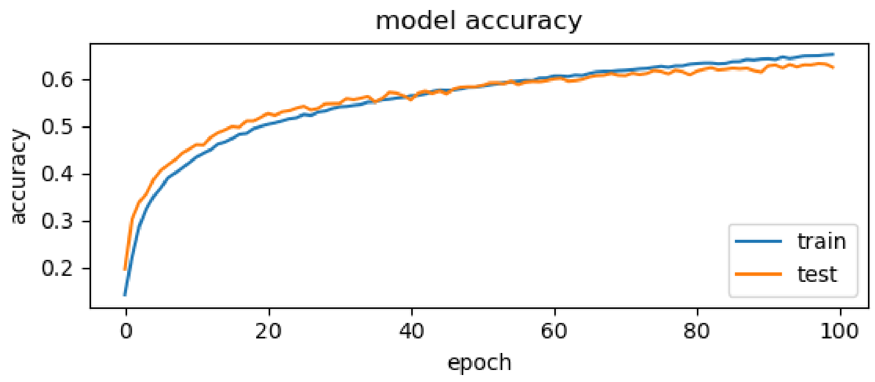

model accuracy
accuracy
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
epoch
train
test

Fig. 49

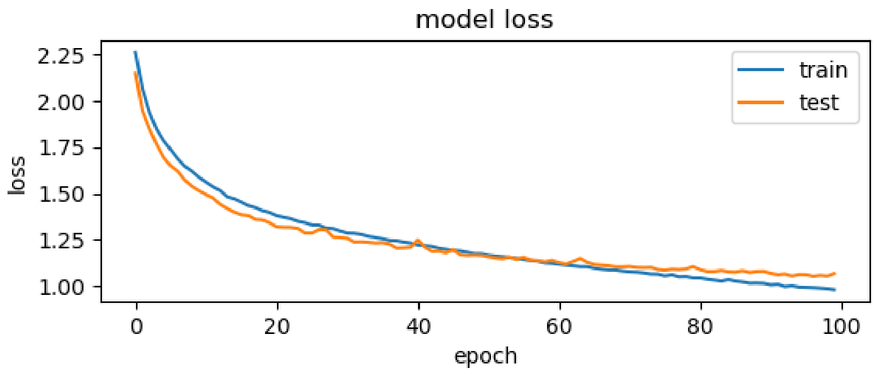

model loss
loss
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100
epoch
train
test

Fig. 50

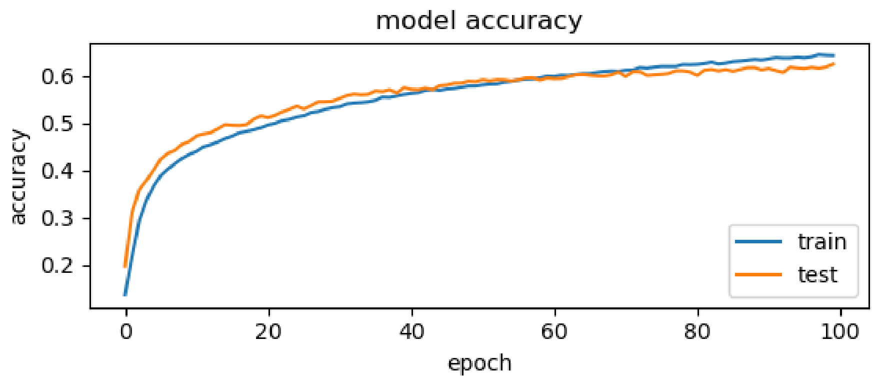

model accuracy
accuracy
epoch
train
test
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

Fig. 51

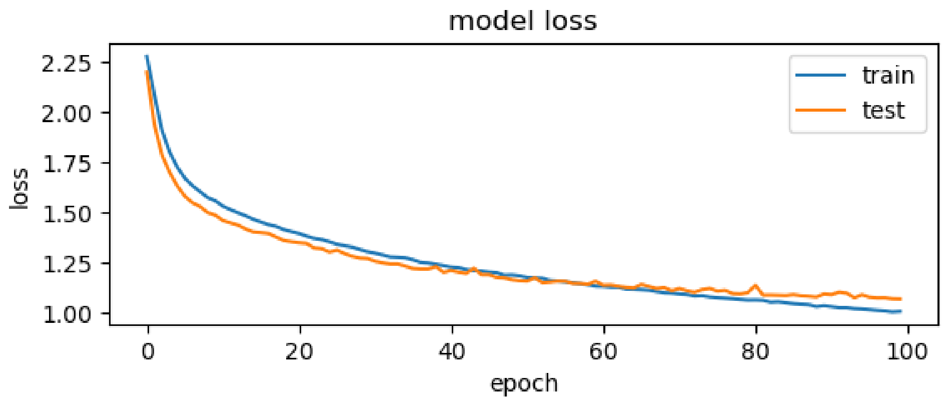

model loss
loss
epoch
train
test
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100

Fig. 52

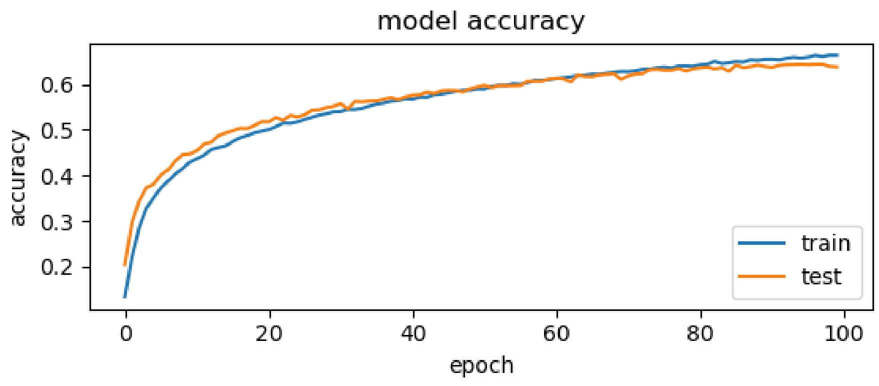

model accuracy
accuracy
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
epoch
train
test

Fig. 53

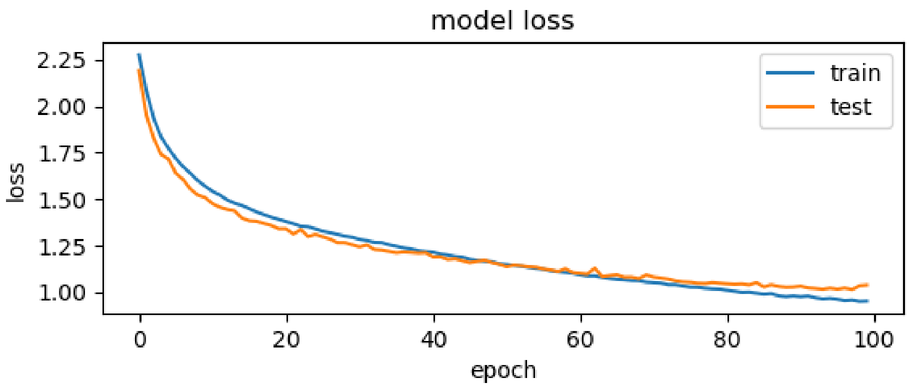
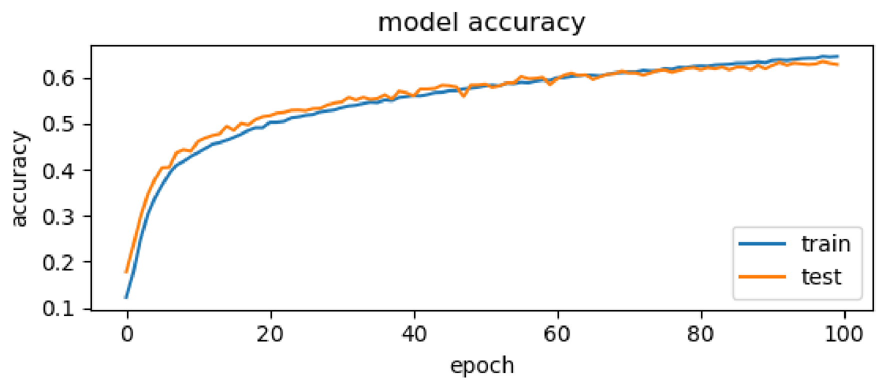
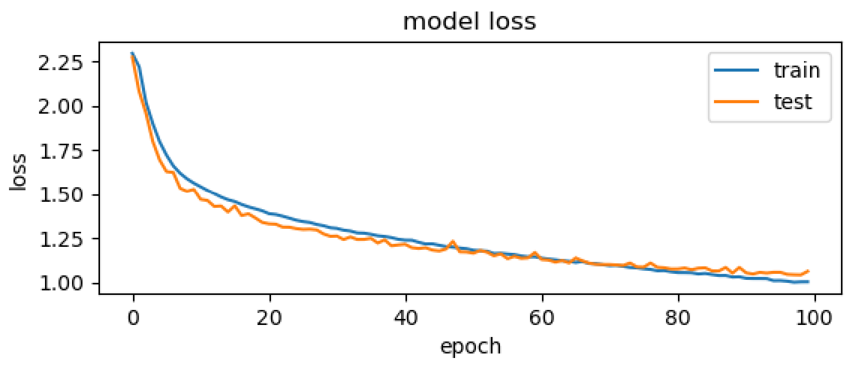

model loss
loss
2.25
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100
epoch
train
test

Fig. 54


model accuracy
accuracy
epoch
train
test
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

<u>Fig. 55</u>


model loss
loss
epoch
train
test
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100

<u>Fig.56</u>

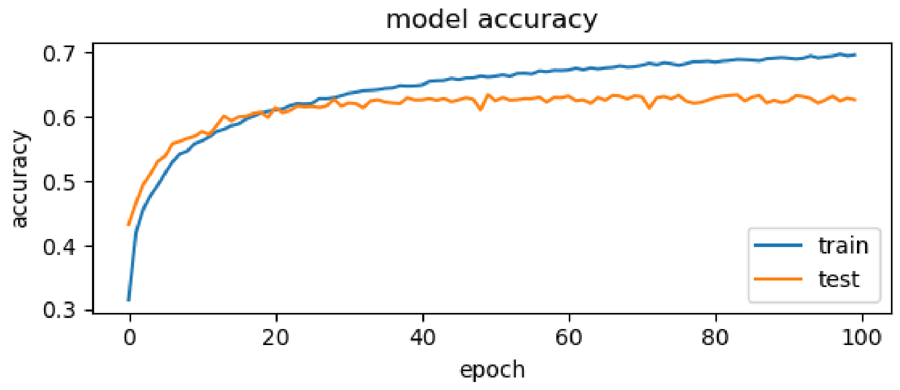

model accuracy
accuracy
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
epoch
train
test

Fig. 57

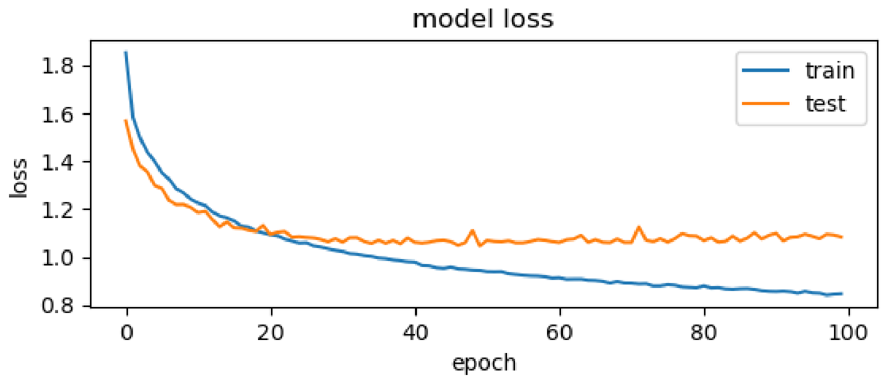

model loss
loss
0.8
1.0
1.2
1.4
1.6
1.8
0
20
40
60
80
100
epoch
train
test

Fig. 58

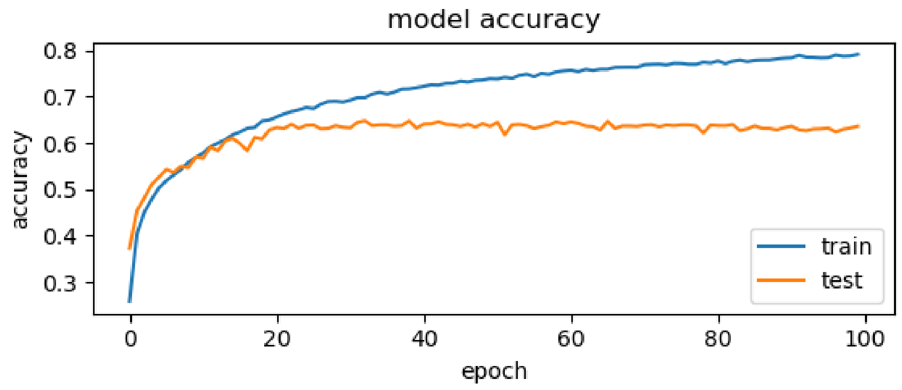

model accuracy
accuracy
0.8
0.7
0.6
0.5
0.4
0.3
train
test
0
20
40
60
80
100
epoch

Fig.59

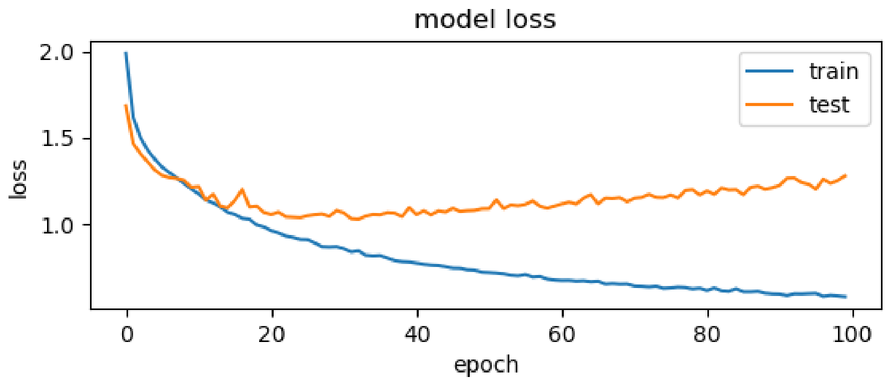

model loss
loss
2.0
1.5
1.0
train
test
0
20
40
60
80
100
epoch

Fig. 60

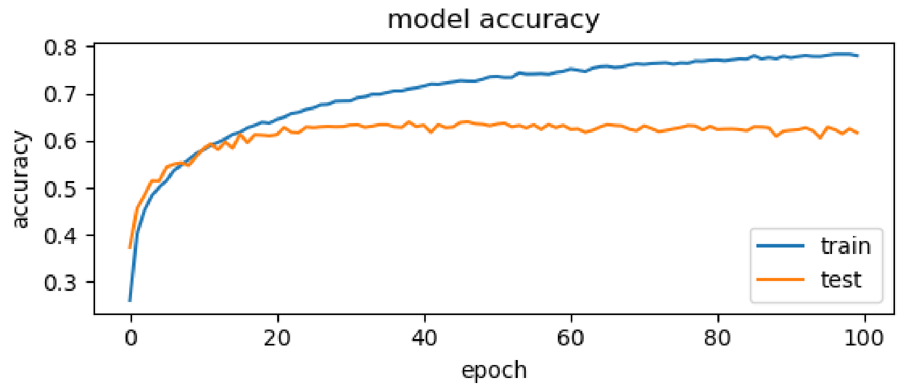

model accuracy
accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 61

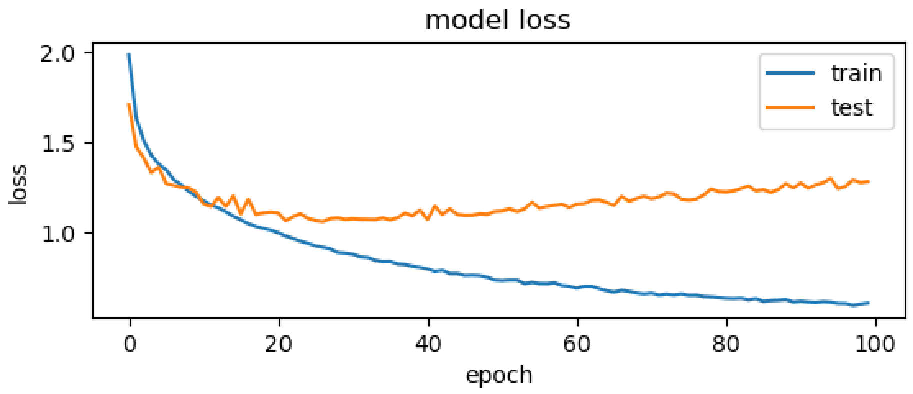

model loss
loss
2.0
1.5
1.0
0
20
40
60
80
100
epoch
train
test

Fig.62

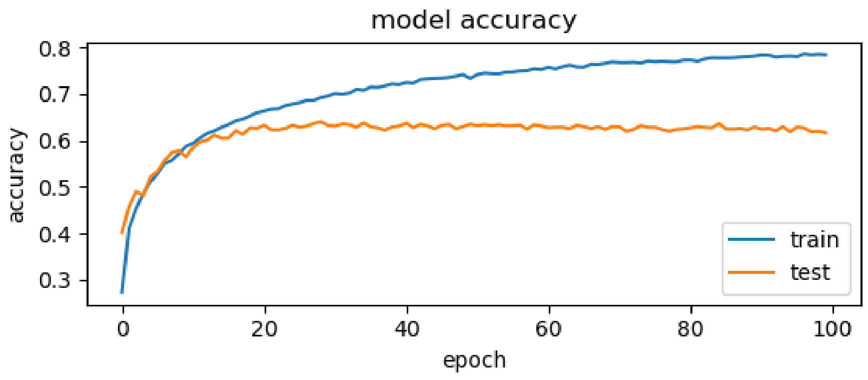

model accuracy
accuracy
epoch
train
test
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Fig. 63

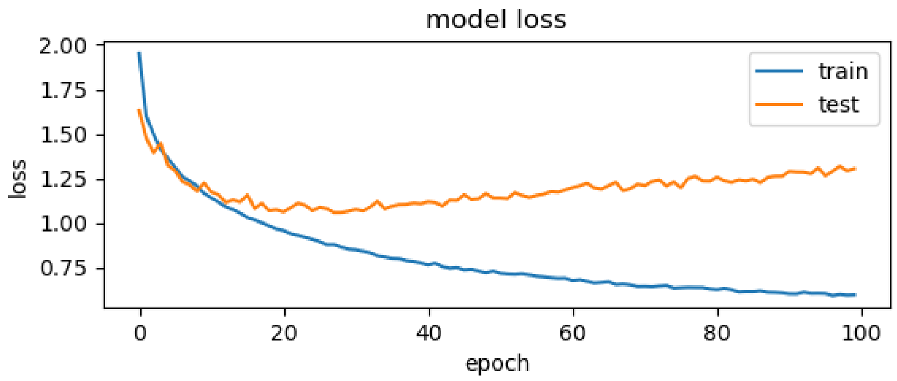

model loss
loss
epoch
train
test
2.00
1.75
1.50
1.25
1.00
0.75
0
20
40
60
80
100

Fig.64

iii. Adam: (beta_1 = 0.9 and beta_2 = 0.999)

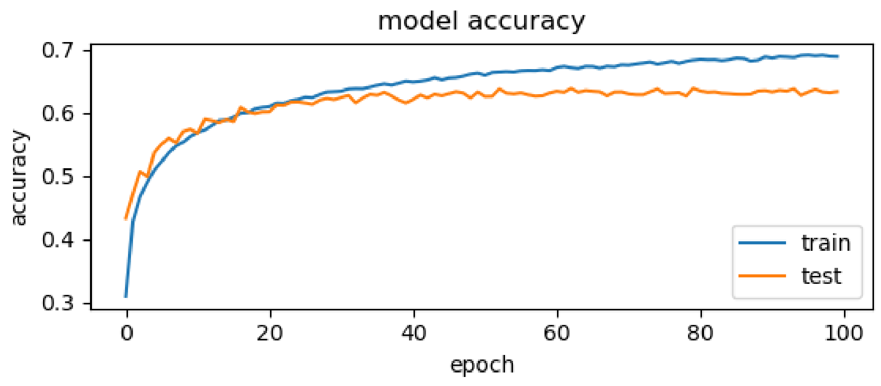


Fig. 65

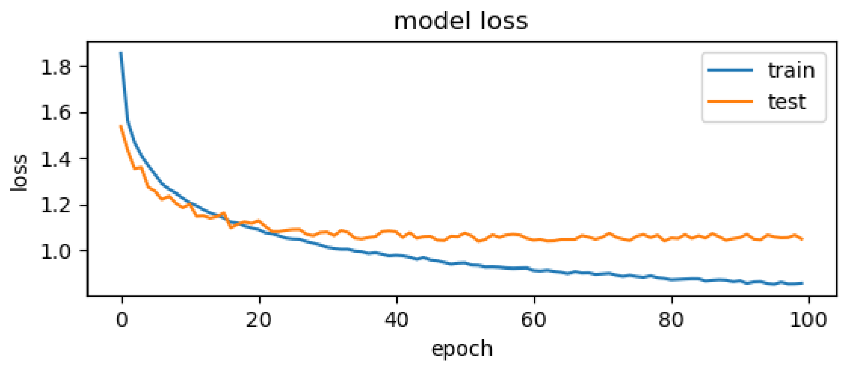
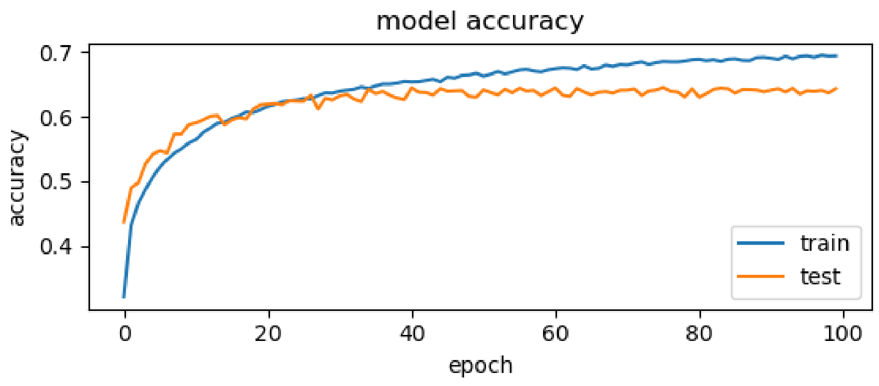
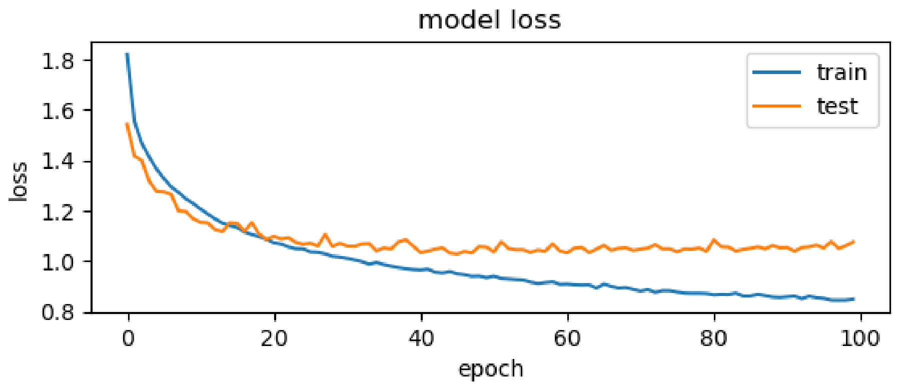


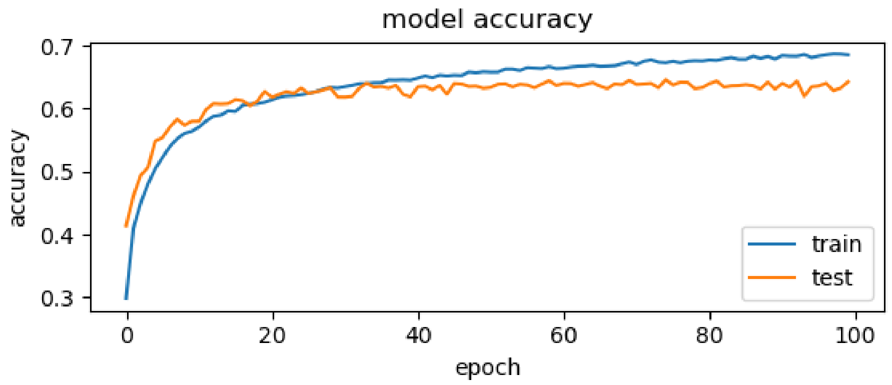
Fig. 66


model accuracy
accuracy
0.7
0.6
0.5
0.4
epoch
0
20
40
60
80
100
train
test

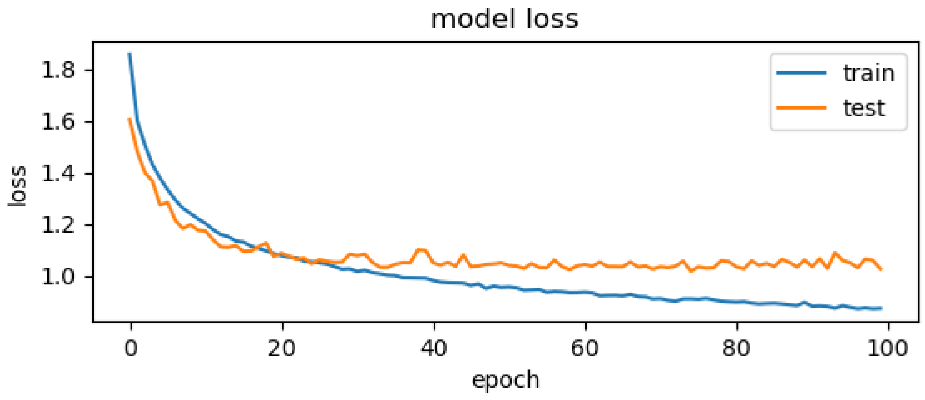
<u>Fig. 67</u>


model loss
loss
1.8
1.6
1.4
1.2
1.0
0.8
epoch
0
20
40
60
80
100
train
test

<u>Fig. 68</u>


model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

<u>Fig. 69</u>


model loss
loss
epoch
train
test
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

<u>Fig. 70</u>

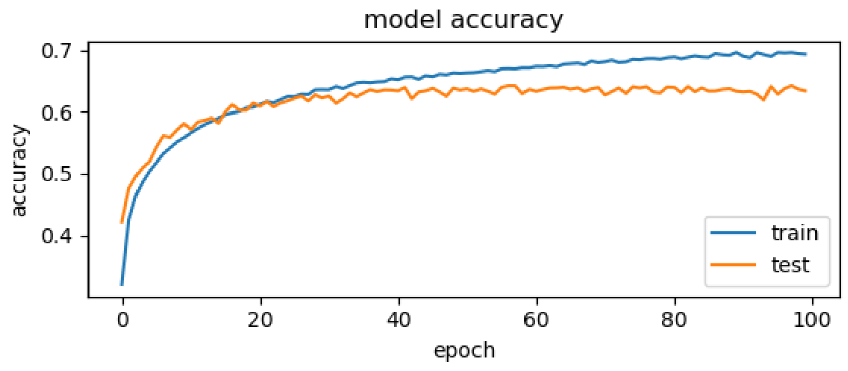

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0
20
40
60
80
100

Fig. 71

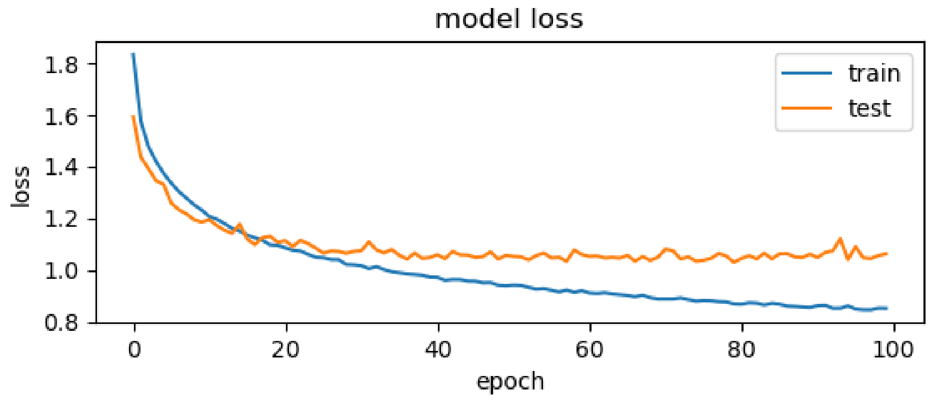

model loss
loss
epoch
train
test
1.8
1.6
1.4
1.2
1.0
0.8
0
20
40
60
80
100

Fig. 72

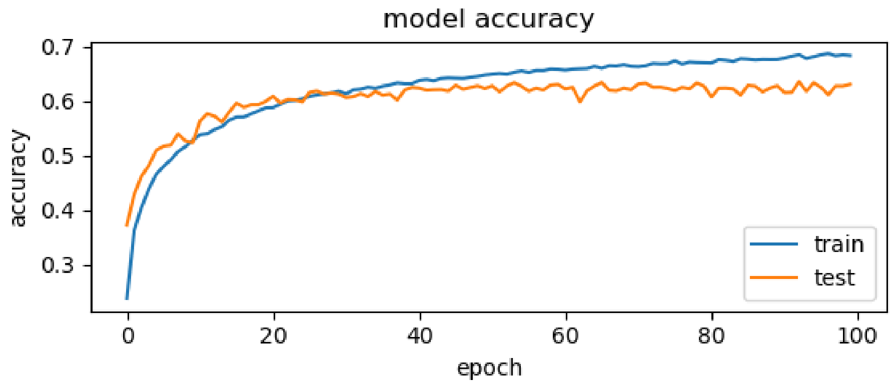

model accuracy
accuracy
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 73

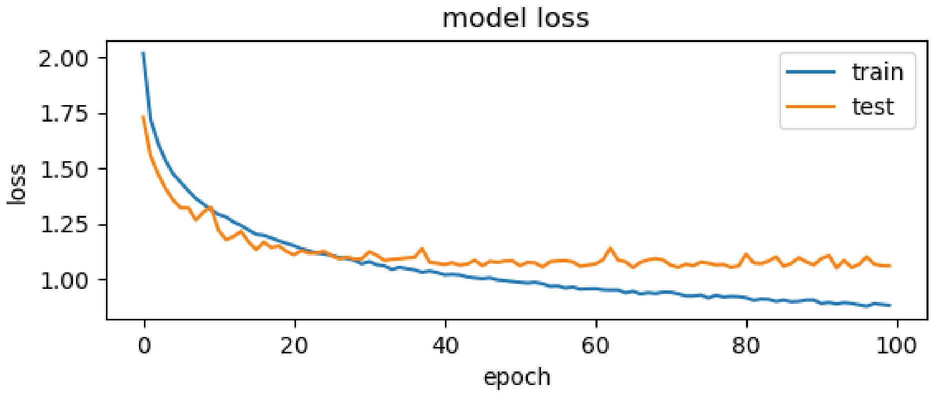

model loss
loss
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100
epoch
train
test

Fig. 74

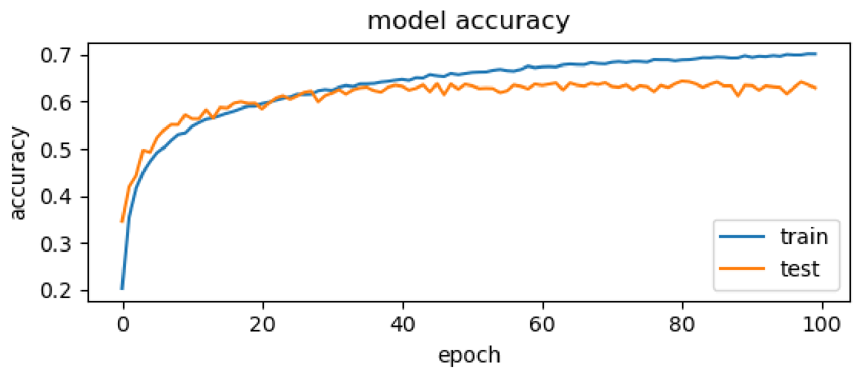

model accuracy
accuracy
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
epoch
train
test

Fig. 75

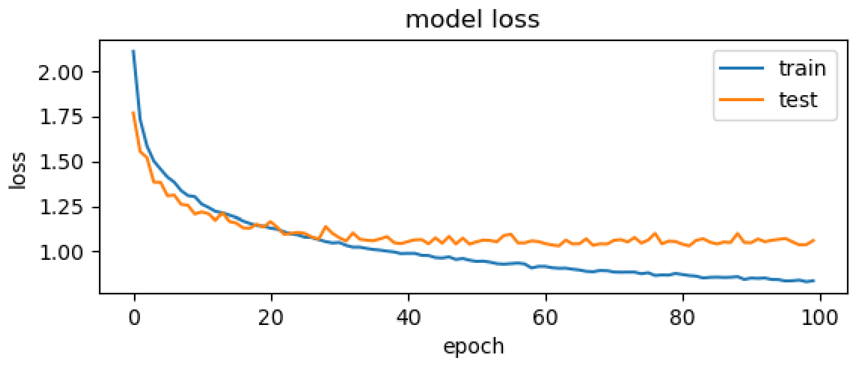

model loss
loss
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100
epoch
train
test

Fig. 76

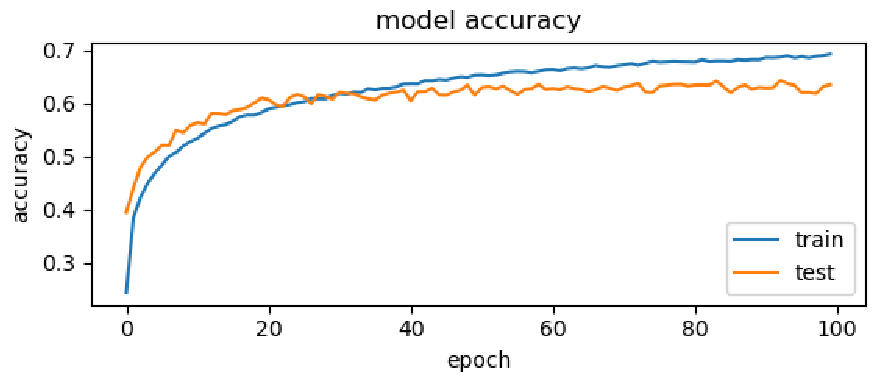

model accuracy
accuracy
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 77

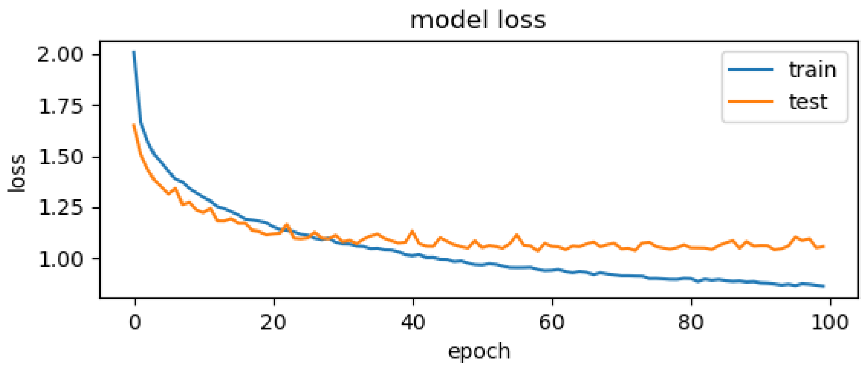
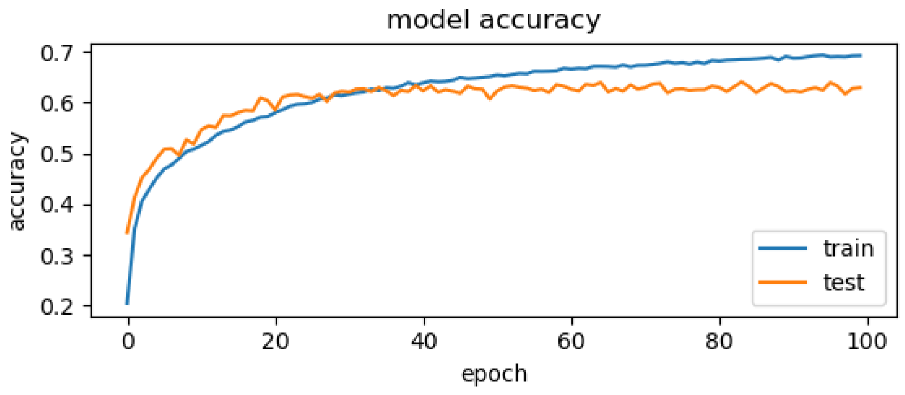
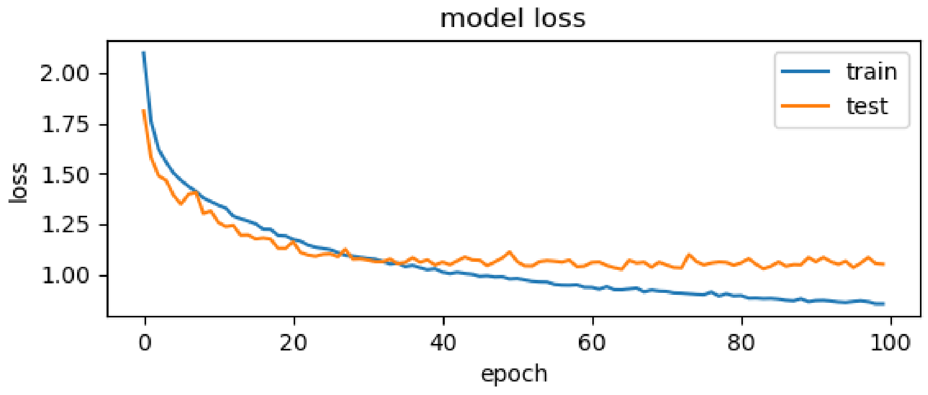

model loss
loss
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100
epoch
train
test

Fig. 78


model accuracy
accuracy
epoch
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
train
test

<u>Fig. 79</u>


model loss
loss
epoch
1.00
1.25
1.50
1.75
2.00
0
20
40
60
80
100
train
test

<u>Fig. 80</u>

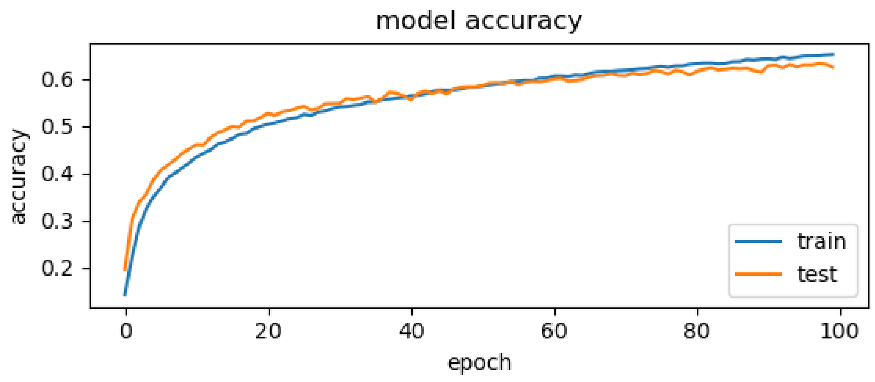

model accuracy
accuracy
epoch
train
test
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

Fig. 81

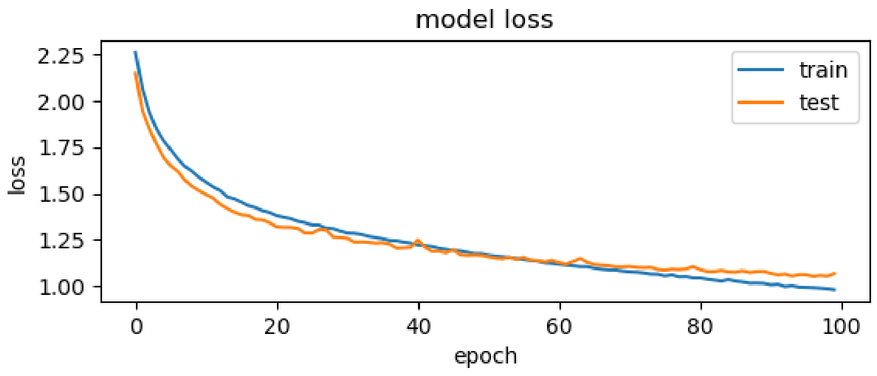

model loss
loss
epoch
train
test
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100

Fig. 82

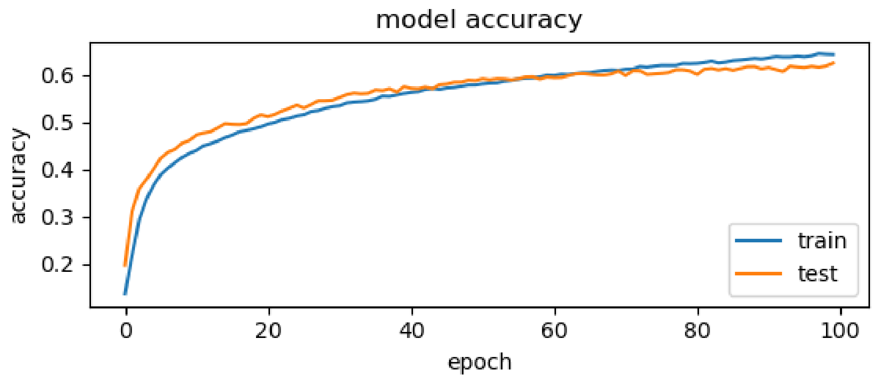

model accuracy
accuracy
0.2
0.3
0.4
0.5
0.6
epoch
0
20
40
60
80
100
train
test

Fig. 83

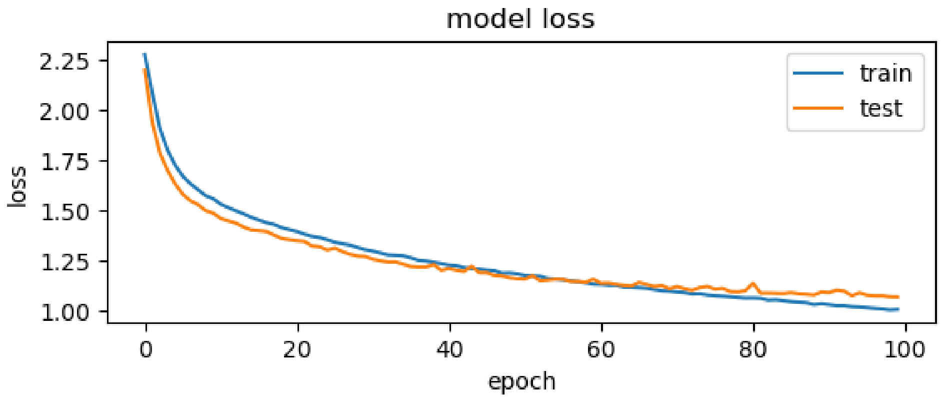

model loss
loss
1.00
1.25
1.50
1.75
2.00
2.25
epoch
0
20
40
60
80
100
train
test

Fig. 84

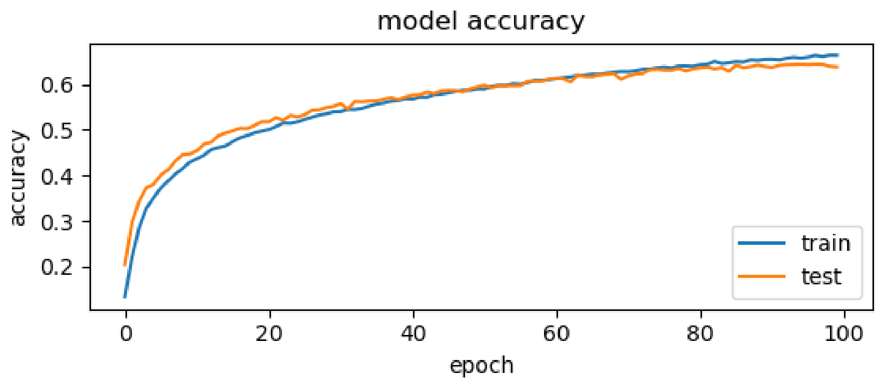

model accuracy
accuracy
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
epoch
train
test

Fig. 85

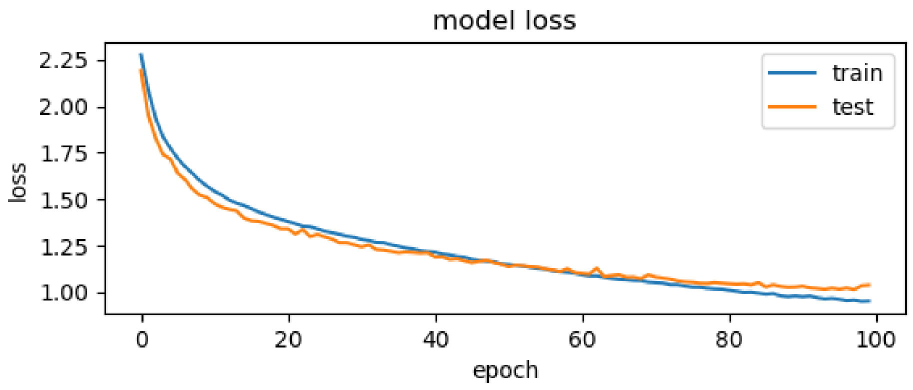

model loss
loss
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100
epoch
train
test

Fig. 86

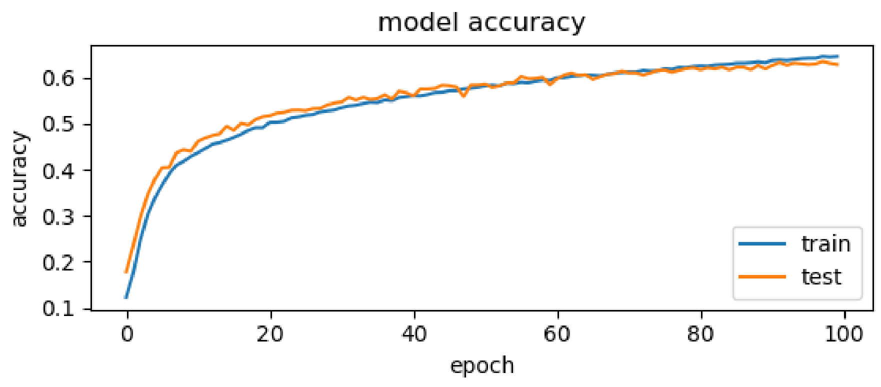

model accuracy
accuracy
epoch
train
test
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

Fig. 87

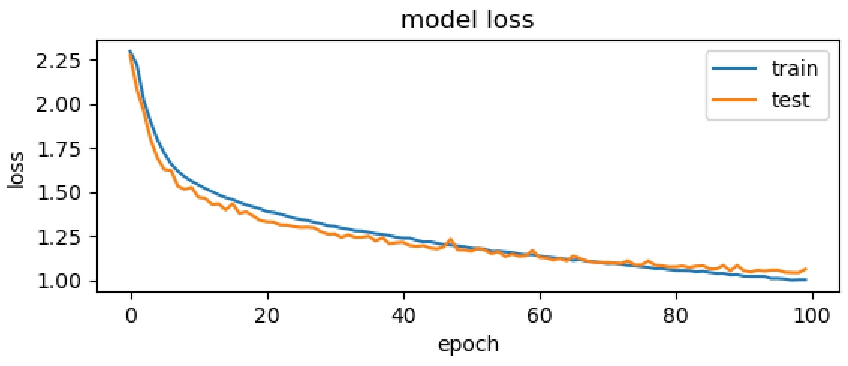

model loss
loss
epoch
train
test
1.00
1.25
1.50
1.75
2.00
2.25
0
20
40
60
80
100

Fig.87

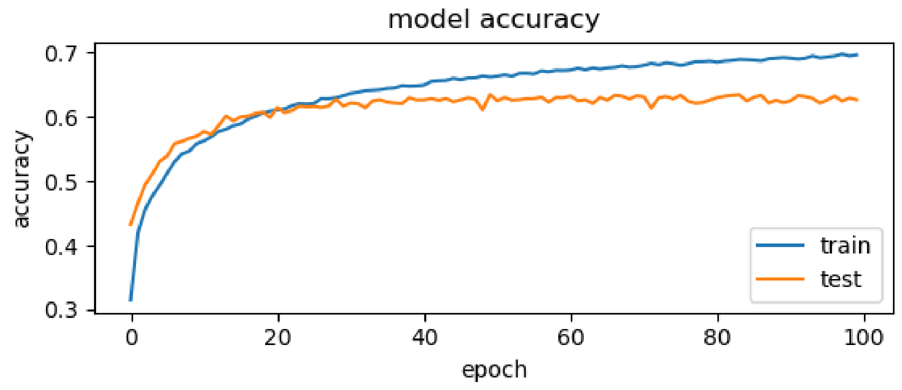

model accuracy
accuracy
epoch
train
test
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Fig. 88

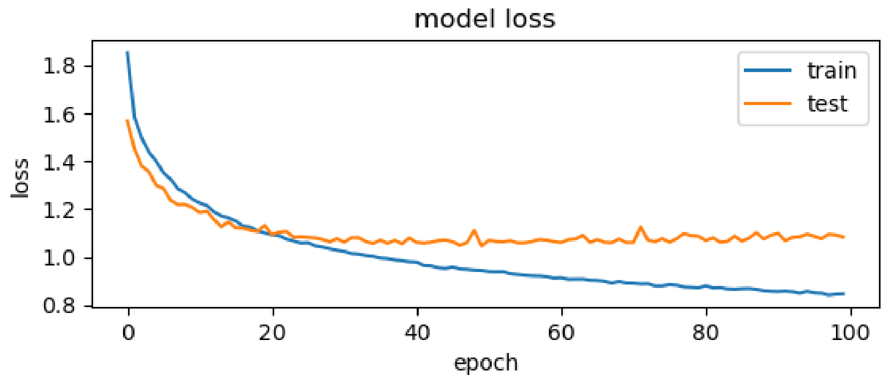

model loss
loss
epoch
train
test
1.8
1.6
1.4
1.2
1.0
0.8
0
20
40
60
80
100

Fig. 89

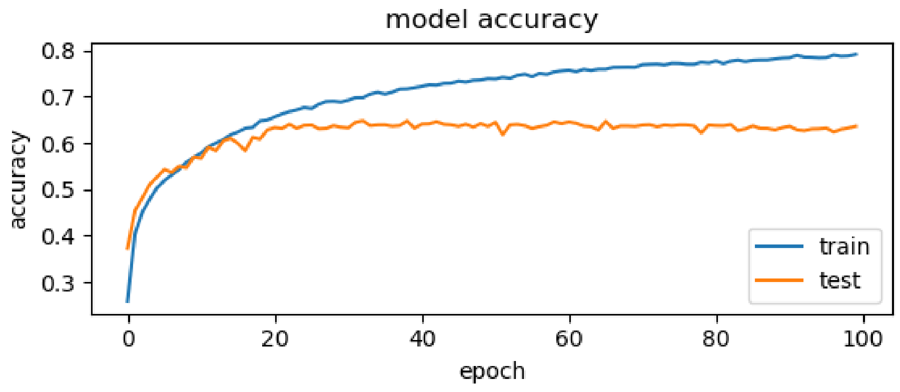

model accuracy
accuracy
epoch
train
test
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100

Fig.90

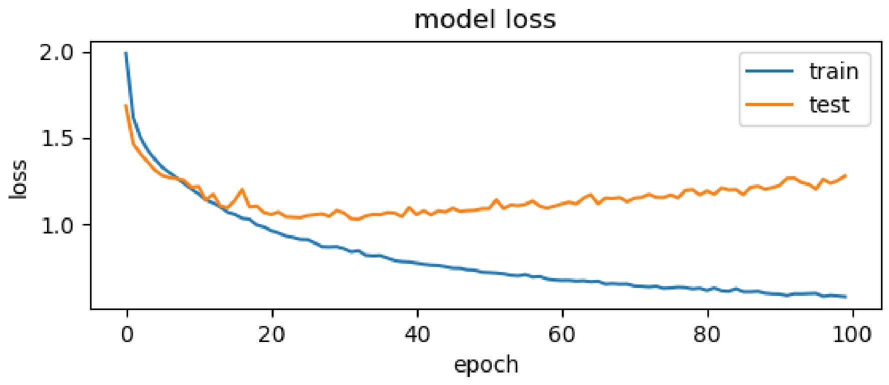

model loss
loss
epoch
train
test
1.0
1.5
2.0
0
20
40
60
80
100

Fig. 91

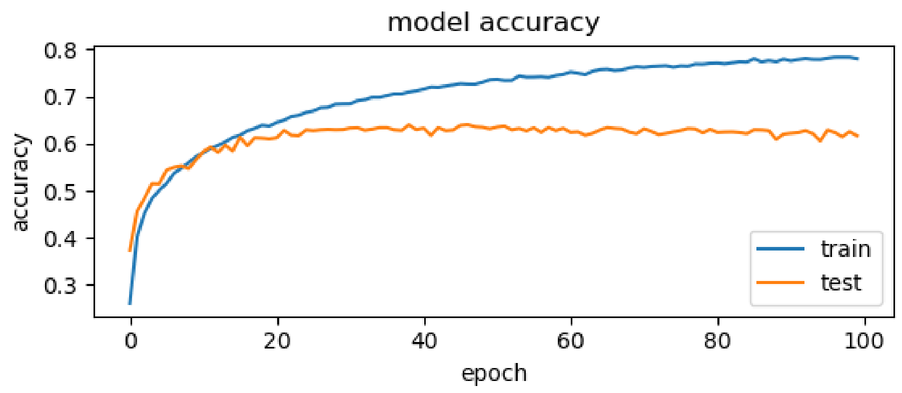

model accuracy
accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 92

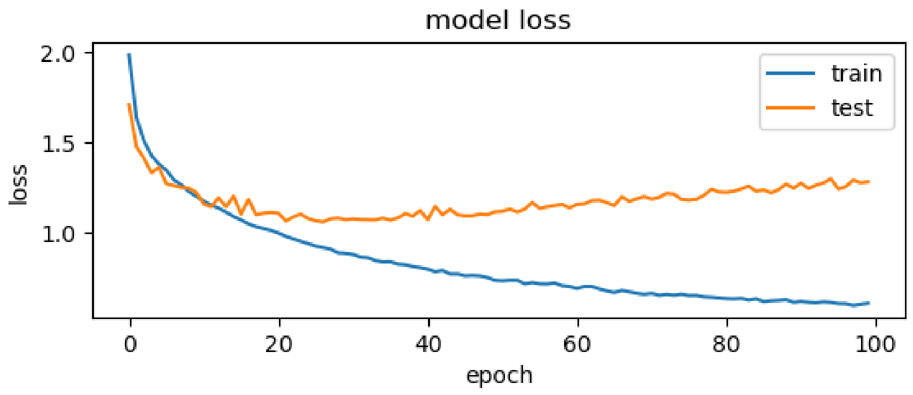

model loss
loss
2.0
1.5
1.0
0
20
40
60
80
100
epoch
train
test

Fig.93

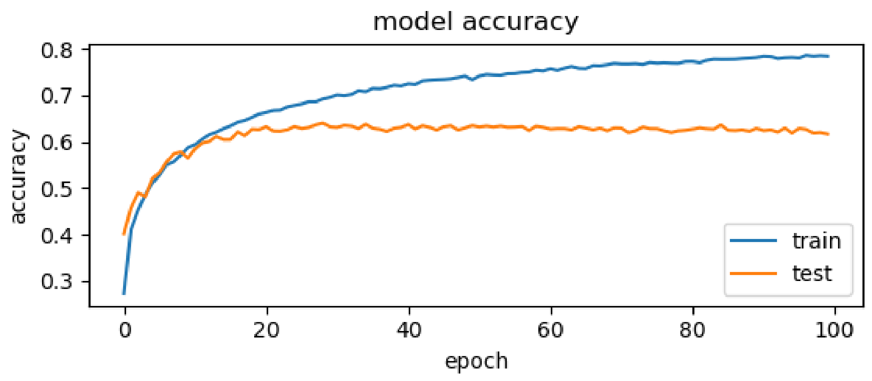

model accuracy
accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
epoch
train
test

Fig. 94

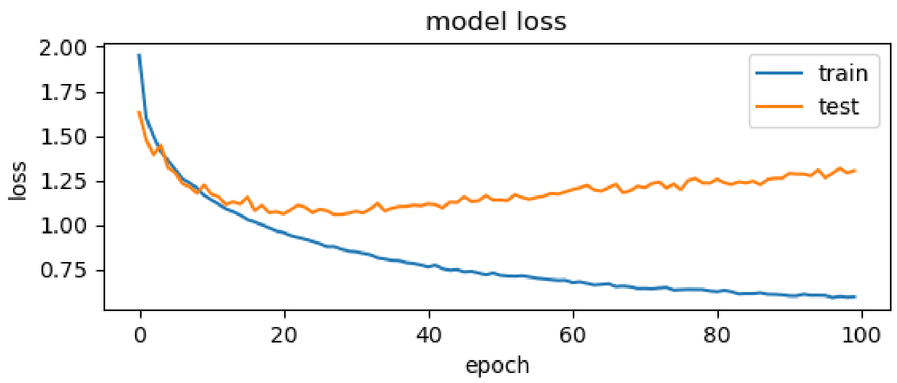

model loss
loss
2.00
1.75
1.50
1.25
1.00
0.75
0
20
40
60
80
100
epoch
train
test

Fig.95

4. DISCUSSION

- **Preprocessing:** Preprocessing step basically includes the normalization step which means dividing the data by 255 in order to get the pixel values in the range of 0 to 1. In this step data is zero-centered by subtracting mean of the entire image from each pixel value to prevent the high frequency component in an image.

- **Random weight initialization:** This step is done after the construction of the network and the weights are initialized to a suitable value at every layer. In this problem statement I have used Xavier weight initialization which is used to maintain the gradient scale similar in all layers. The 2 types of weight initializations used here are Normal and Uniform distributions.

- **Batch-size:** This is one of the most important parameters which decides the performance accuracy and the computation time of the entire neural network. The more is the batch size the more unstable is the network and yields poor performance accuracy and high computation time. The batch size parameter contributes towards the convergence as well as the performance measure of the convolutional neural network. Optimum batch size = 128, which is determined after various experiments performed.

- **Learning Rate:** Learning rate is the parameter which is responsible for determining the convergence time of the neural network. The smaller the learning rate the more is the performance accuracy, and more is the computation time while in the other hand if the learning rate is small then the convergence happens slowly while the larger learning rate doesn’t converge to global minima. Here upon experiments it is observed that the optimum learning rate is 0.001.

- **Optimizer:** The optimizers used here are RMSprop, SGD (Stochastic Gradient Descent) and Adam optimizer. Out of the 3 optimizers used the Adam optimizer proved to be the best and gives high performance accuracy and better outputs out of the 3 optimizers.

- **Epoch:** Epoch is basically defined as the number of repetitions of program or process for a fixed number of times in order to train the image dataset. The technique of backward propagation, cost function, forward propagation and weight updating is known as epoch. Here, I am using 100 epochs for training the model.

- **Dropout:** The basic idea behind the dropout layer is eliminate the unwanted or less priority neurons from the network during a batch in order to reduce the overfitting issue because due to overfitting issue the strong features take control over the weak features. In this model I have used dropout layer after the first fully connected dense layer with 0.25 and 0.5 probabilities. The probability value chosen for the dropout layer makes a remarkable difference towards the computation of the performance accuracy.

We can infer from the table that for training accuracy in decreasing order is as Adam, SGD and RMSprop.
We can infer from the table that for testing accuracy in decreasing order is as RMSprop, Adam and SGD.

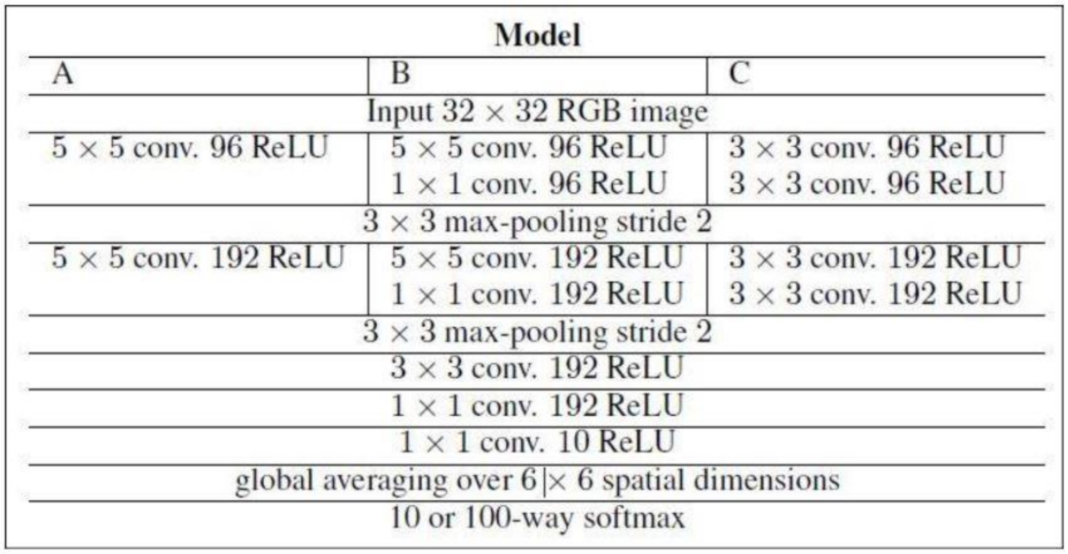
(c) State-of-the-Art CIFAR-10 Classification:

1. ABSTRACT & MOTIVATION

Here, in this problem the task is to study any one of the papers from the list and then compare the approaches taken by the author and improvement done in the CNN for the CIFAR-10 dataset in order to achieve the high-performance accuracy. The experiments were carried out using the high-end processors and GPU's to achieve better results with high amount of computations and less computation time.
Upon reviewing few papers, I choose to analyze the techniques and the improvements done in the paper titled as "*STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET*" in order to compare my results with the results presented in the paper.

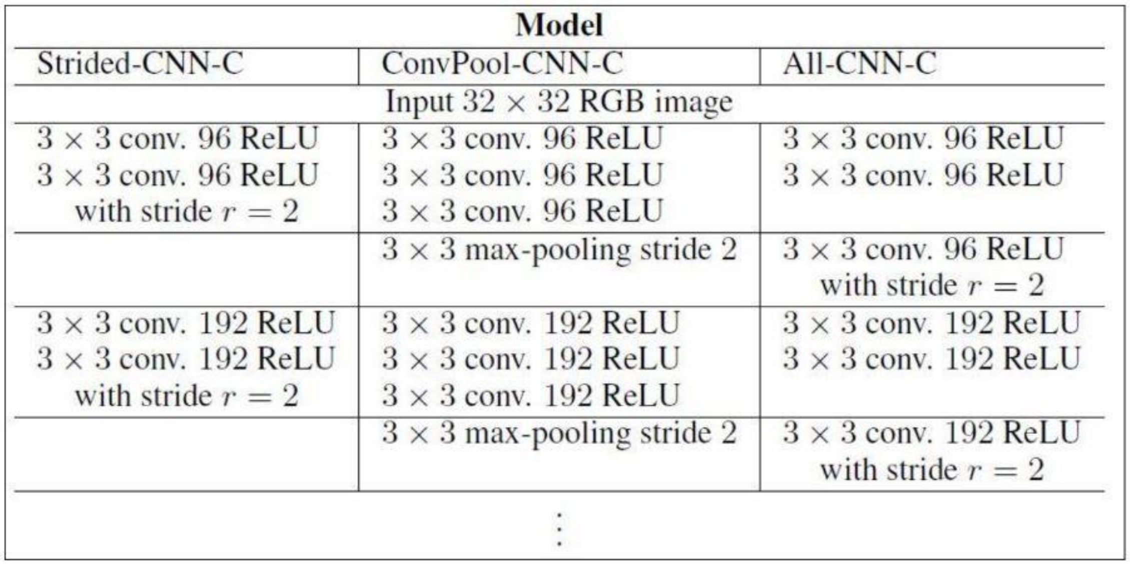
2. DISCUSSION

i. Here, in this particular given paper the authors have adopted the Spatial Convolutional layer using 3 various architectures in order to achieve the high-performance accuracy. The 3 models chosen i.e., A, B and C can be shown as follows:



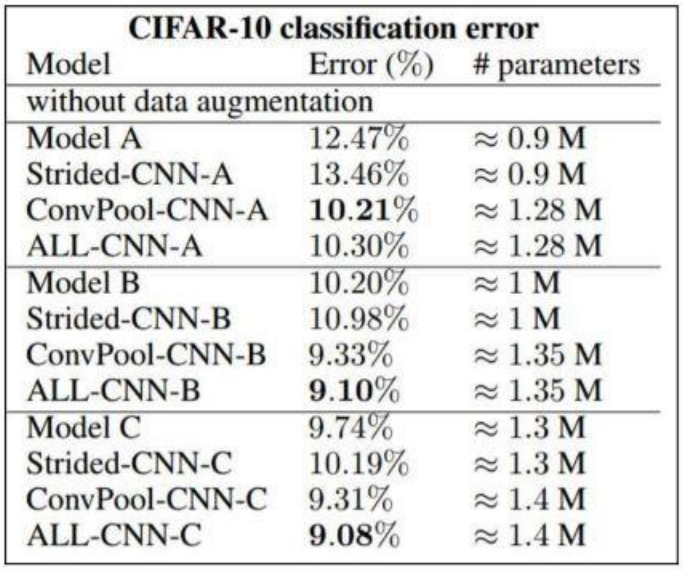
Model		
A	B	C
Input 32 × 32 RGB image		
5 × 5 conv. 96 ReLU	5 × 5 conv. 96 ReLU 1 × 1 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU
3 × 3 max-pooling stride 2		
5 × 5 conv. 192 ReLU	5 × 5 conv. 192 ReLU 1 × 1 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
3 × 3 max-pooling stride 2		
3 × 3 conv. 192 ReLU		
1 × 1 conv. 192 ReLU		
1 × 1 conv. 10 ReLU		
global averaging over 6 × 6 spatial dimensions		
10 or 100-way softmax		

As we can make out from the table given above that the amount of parameters increases, starting from model A to the model C. In all the networks in order to maintain the same level of comparison to generate the 10 output results using a softmax layer to generate the probability of classes obtained by computing the average overall. The additional tasks were also performed to the fully connected layers in substitution of the 1x1 convolutions but later realized that they downgraded their accuracy from 0.5% to 1% than the fully connected convolutional layers.
Moving on to model B, here the improvisation done is that only single 1x1 convolution is carried out following after every normal convolutional layer.
In model C, there is just the replacement of every 5x5 convolutions with all 3x3 convolutions in order to fulfill the 2 main purposes which are as, firstly, the architecture unification which will consist just 3x3 spatial neighbors of previous feature maps and secondly, if the replacement of the max-pooling layer is done with the convolutional layer then the minimum kernel size will be 3x3 which will have stride parameter of 2, allowing the overlapping convolution.
Additionally, the 3 more variants of model C were derived displaying the importance of the pooling required for an image classification.



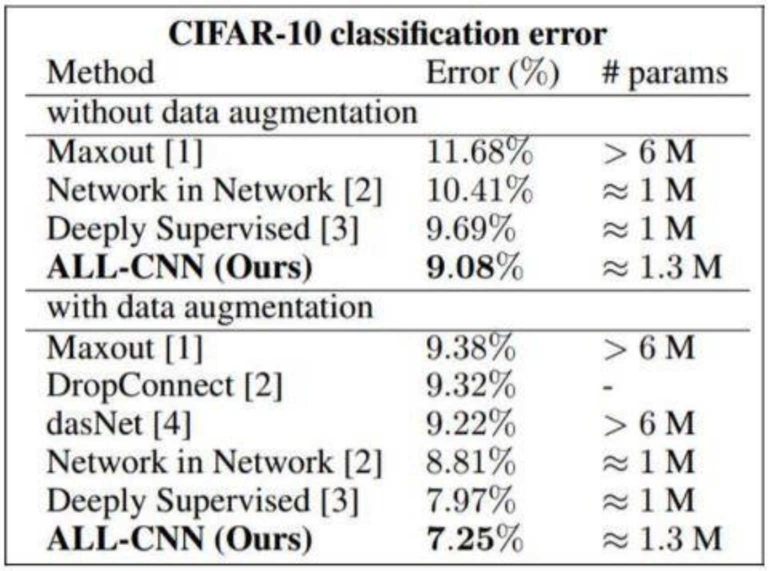
Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
	Input 32 × 32 RGB image	
3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU with stride $r = 2$	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 96 ReLU with stride $r = 2$
3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU with stride $r = 2$	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 192 ReLU with stride $r = 2$
	⋮	

In the "*Strided-CNN-C*" we can observe that the max-pooling layer is eliminated and also the stride parameter of the convolution layer before the max pool layers in raised by 1 in order to ensure that the succeeding layers have the same spatial region.
In the "*All-CNN-C*" model the max-pooling layers is replaced by the convolutional layer.
In the "*ConvPool-CNN-C*" model a heavy convolution is incorporated before every max pooling layer having the same kernel size as that of the pooling layer.
The results obtained in the paper are given below:



CIFAR-10 classification error

Model	Error (%)	# parameters
without data augmentation		
Model A	12.47%	≈ 0.9 M
Strided-CNN-A	13.46%	≈ 0.9 M
ConvPool-CNN-A	**10.21%**	≈ 1.28 M
ALL-CNN-A	10.30%	≈ 1.28 M
Model B	10.20%	≈ 1 M
Strided-CNN-B	10.98%	≈ 1 M
ConvPool-CNN-B	9.33%	≈ 1.35 M
ALL-CNN-B	**9.10%**	≈ 1.35 M
Model C	9.74%	≈ 1.3 M
Strided-CNN-C	10.19%	≈ 1.3 M
ConvPool-CNN-C	9.31%	≈ 1.4 M
ALL-CNN-C	**9.08%**	≈ 1.4 M



CIFAR-10 classification error

Method	Error (%)	# params
without data augmentation		
Maxout [1]	11.68%	> 6 M
Network in Network [2]	10.41%	≈ 1 M
Deeply Supervised [3]	9.69%	≈ 1 M
ALL-CNN (Ours)	**9.08%**	≈ 1.3 M
with data augmentation		
Maxout [1]	9.38%	> 6 M
DropConnect [2]	9.32%	-
dasNet [4]	9.22%	> 6 M
Network in Network [2]	8.81%	≈ 1 M
Deeply Supervised [3]	7.97%	≈ 1 M
ALL-CNN (Ours)	**7.25%**	≈ 1.3 M

ii. Comparing the solution obtained with LeNet-5 and the approaches adopted in the paper I can list few comparisons based on my analysis:

Sr. No.	**LeNet-5**	**CNN of paper**
1.	The architecture used here consists of small numbers of filters and neurons.	The architecture used here consists of large numbers of filters and neurons.
2.	The complexity of model is less.	The complexity of the model is high. It has 7 convolutional layers, 2 Max-Pooling layers & 1 global averaging layer. Model-C has 3 different

		variations thereby making it more complex.
3.	Around 100 epochs were used	200 – 350 epochs were used.
4.	Normal CPU can be used for computation	High performance GPU will be required for computations
5.	Learning rate implemented is 0.01 & 0.001	Learning rate implemented was 0.25, 0.1, 0.05, 0.01 & 0.001
6.	Testing accuracy obtained is in the range of 60-67%	Testing accuracy obtained is in the range of 85-97%
7.	There are cases when sometimes the testing accuracy is greater than the training accuracy.	The training accuracy is always greater than the testing accuracy.
8.	The range of values for the dropout layer is from 0.25 – 0.7.	The range of values for the dropout layer is from 0.2 – 0.5.
9.	The kernel size is 5x5, stride parameter is 2 and the non-linear activation function used is ReLU.	Various kernel sizes such as 5x5, 3x3 or 1x1 are used, stride parameter is 2 and the non-linear activation function used is ReLU.
10.	It requires less computation time.	It requires very high computation time.

References:

1. Wikipedia
2. Deep learning CNN Convolutional Neural Network LeNet-5 detailed explanation
3. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
4. STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET Jost Tobias Springenberg∗ , Alexey Dosovitskiy∗ , Thomas Brox, Martin Riedmiller
5. Tutorialspoint.com (Keras)
6. Keras.io