# Implementing a feedforward which architecture which accounts for rapid categorization(Hmax model)

**Mohammad Mohammad Beigi**[a]

[a]Student, Department, Sharif University of Technology

This manuscript was compiled on July 8, 2023

**Primates have an exceptional ability to recognize objects, and despite significant advancements in computer vision systems, primates still surpass them in terms of visual capabilities and resilience to image degradation. Primates excel at quickly categorizing objects and processing visual stimuli, particularly in tasks that involve rapid presentation of images. This remarkable visual processing ability is likely achieved through a feedforward mechanism, considering the multiple stages of processing and neural delays involved.**

**In this homework, we run the proposed model of the main resource of the homework to show that a specific implementation of feedforward theories of object recognition(Hmax), which is based on the hierarchical organization of cells proposed by Hubel and Wiesel [2] and takes into account anatomical and physiological constraints, can accurately predict human performance in a task where participants rapidly categorize masked animal versus non-animal stimuli in terms of both level and pattern.**

Keywords: Hmax | sumfilter | maxfilter | SVM | d'

**HMAX** [1] model is a computational model of visual recognition developed by Tomaso Poggio and his colleagues at MIT. It is inspired by the hierarchical organization and processing in the visual cortex of the brain.

The HMAX model aims to explain how the human visual system processes and recognizes objects. It consists of multiple stages that mimic different levels of visual processing. Here are the main stages of the model:

1. Retina: The input to the model is an image, which is first processed by simulating the responses of cells in the retina, such as center-surround cells.

2. S1 layer: This layer represents simple cells, which are modeled as linear filters that respond to specific orientations and spatial frequencies. The responses of these cells are calculated using a technique called Gabor filtering.

3. C1 layer: This layer represents complex cells, which are modeled as pooling units that combine the responses of nearby S1 cells with similar orientation selectivity. This pooling operation helps to achieve position invariance, meaning the model can recognize objects regardless of their location in the visual field.

4. S2 layer: This layer represents cells with larger receptive fields and responds to combinations of local features from the C1 layer. The S2 layer is designed to capture more complex and invariant features.

5. C2 layer: This layer performs further pooling of the S2 responses and introduces more invariance to small transformations, such as changes in position, scale, and rotation.

6. IT layer: This layer represents cells in the inferotemporal cortex, which are known to be involved in object recognition.[3-4-5] The IT layer combines responses from multiple C2 units to form a high-level representation of the input image, which can be used for categorization or identification.

The HMAX model has been successful in explaining several aspects of visual processing, including the emergence of selectivity to different features at each stage and the ability to generalize and recognize objects under various transformations. However, it is worth noting that the model is a simplified representation of the visual system and does not capture all the complexities and intricacies of human vision.

## Materials and Methods

**Dataset.** We utilized a stimulus database consisting of 600 animal stimuli and 600 non-animal stimuli. The animal images were classified into four categories, namely head, close-body, medium-body, and far-body, each containing 150 examples. On the other hand, the non-animal stimuli were selected from a database of annotated mean-depth images and served as distractors. To ensure consistency, all images in the database were grayscale and had a resolution of 256 x 256 pixels.

**Categorization by the Model.** In order to train the PFC (prefrontal cortex) classification unit in our model, we implemented a procedure called random splits, which is known for its reliability in estimating classifier performance. This procedure consisted of the following steps:

1. We had a set of 1,200 images, including both animal and non-animal stimuli. This set was divided equally into two halves: one was labeled as the "training" set, and the other as the "test" set.

2. From the training set, we randomly selected specific examples of animal and non-animal images at a rate of 25%. These selected examples were used to imprint S4 units. S4 units are units in lower stages that become tuned to patches of natural images. By storing the activity patterns of their inputs during the presentation of a specific example, S4 units acquire tuning specific to views of the target object. This observation aligns with previous evidence that suggests neurons in the inferior temporal cortex (IT) exhibit selectivity influenced by visual experiences.

3. In the context of a classification unit, the relationship between the input pattern x and the output y can be expressed using the equation $y = \sum c_j \times x_j$. Here, $c_j$ represents the weight assigned to the j-th input, and $x_j$ represents the value

Please provide details of author contributions here.

[1]A.O.(Author One) and A.T. (Author Two) contributed equally to this work (remove if not applicable).

of the j-th input. To obtain a classification label, the output y is converted into either -1 or 1. In the supervised learning phase, the synaptic weights c are adjusted to minimize the overall classification error E when applied to the training set. The objective is to train the PFC classification unit using a labeled "training" set of images and assess its performance on a separate "test" set.

4. The performance of the classifier on the "test" set was assessed. We conducted the entire procedure 1 time.(each run takes 2.5 hours)

**Functions.**

- The demoRelease function showcases the utilization of C2 standard model features within a framework for pattern classification.
- The readAllImages function reads all training and testing images and stores them in a cell of length 4. The images are organized as follows: - cI1 contains the positive training set.
  - cI2 contains the negative training set.
  - cI3 contains the positive testing set.
  - cI4 contains the negative testing set.
- The extractRandC1Patches function randomly selects prototypes during the training process of the C2 classification system. Currently, it only extracts prototypes from Band 2. Extracting prototypes from all bands may improve performance.
- The init_gabor function implements a Gabor Filter.
- The extractC2forcell function serves as a wrapper for the C2 function. It takes a cell of images, cImages, as input and extracts the C2 layer values for all prototypes in the cell cPatches.
- The CLSnn function constructs a neural network (NN) classifier.
- The C1 function calculates the activation of C1 and S1 units given an input image, stim.
- The C2 function extracts layers S1, C1, S2, and finally C2 from an input image.
- The WindowedPatchDistance function computes the Euclidean distance between a patch and all crops of images that have similar sizes.

  More important functions are here:

- maxfilter: Performs morphological dilation on a multilayer image. The "max operation" refers to a key component of the HMAX model, which is a computational model of object recognition developed by Tomaso Poggio and his colleagues. HMAX stands for Hierarchical Model and X, where X denotes a specific layer within the model.

  In the HMAX model, the visual processing hierarchy consists of multiple layers that mimic the organization of the human visual system. Each layer performs a specific computation to extract increasingly complex features from the input visual stimuli. The max operation is primarily employed in the early layers of the HMAX model.

  The max operation is a non-linear operation that computes the maximum response within a given local neighborhood of the input image. It helps capture the presence of specific visual features, such as edges or contours, at different positions and orientations. By applying the max operation, the HMAX model can be more robust to variations in the exact location or appearance of these features.

  To illustrate how the max operation works, let's consider a simplified example. Suppose we have a small patch of an image, represented as a 2D array of pixel values:

$$\begin{bmatrix} 2 & 8 & 5 \\ 10 & 3 & 7 \\ 4 & 9 & 1 \end{bmatrix}$$

  In this case, let's assume the receptive field of the max operation is a 3x3 window that can slide over the image. The max operation involves selecting the maximum value within each receptive field, producing a new output array with the maximum values:

$$\begin{bmatrix} 10 & 10 & 9 \\ 10 & 10 & 9 \\ 10 & 10 & 9 \end{bmatrix}$$

  The resulting output emphasizes the regions where the maximum values were found, effectively preserving the most salient features. In subsequent layers of the HMAX model, additional operations are applied to the output of the max operation, leading to a hierarchical representation of increasingly complex features.

  It's important to note that the HMAX model is a simplified model of the visual system and does not capture all the intricacies of human vision. However, it provides insights into how early visual processing stages might contribute to object recognition and has influenced the development of more advanced models in computer vision research.

- sumfilter: The "sum" operation in the HMAX model refers to a pooling operation that is performed at a specific layer of the model. The HMAX model is a computational model of the visual cortex that attempts to simulate the hierarchical processing of visual information.

  During the sum operation, the responses from multiple filter units within a specific receptive field are combined by taking their sum. This pooling operation serves two main purposes:

  1. Spatial Invariance: By summing the responses from different filter units, the pooling operation helps to make the model more robust to small spatial variations in the input image. It allows the model to respond to the presence of a particular feature regardless of its exact location in the visual field.

  2. Dimensionality Reduction: The sum operation also helps in reducing the dimensionality of the feature map. Instead of representing the precise responses of each individual filter unit, the pooling operation condenses the information by summarizing the total response within a

local neighborhood. This reduces the computational complexity and makes subsequent processing more efficient.

Overall, the sum operation in the HMAX model contributes to the model's ability to capture and process visual information in a hierarchical manner, allowing it to recognize complex patterns and objects in an image.

## Results

After learning Procedure we obtain 4 matrices named "XTrain", "XTest", "ytrain" and "ytest" which contains examples and labels of train and test sets.

## 1. First part

In this part We train the data set on Head and Medium body and test on Far and Close body.(Actually at first I divided the images wrongly and ran the model. I thought that it is better if I don't put the created data away )

**Nearest Neighbor Classifier.** We use a Nearest Neighbor Classifier to calculate the accuracy of the predictions of the model. (figure 1)

```
Model = CLSnn(XTrain, ytrain); %training
[ry,rw] = CLSnnC(XTest,Model); %predicting new labels
```

**Fig. 1.** Matlab Code to classify by NN Classifier

```
successrate =

    0.6917
```

**Fig. 2.** accuracy while classifying by NNC

As you see we obtained accuracy 0.6917 by NNC.(figure 2)

**Support Vector Machine Classifier.** We have saved the results of train and test sets in two .csv files. We apply the SVM Classifier by R code in figure 3.

```r
# Create the SVM model
svm_model <- svm(Y ~ ., data = train_data)
summary(svm_model)
# Make predictions on the test data
predictions <- predict(svm_model, newdata = test_data)
# Assign 1 to positive predictions and -1 to negative predictions
predictions <- ifelse(predictions > 0, 1, -1)
# Calculate the sum of differences and divide by 600
sum_diff <- sum(predictions - test_data$Y == 0) / 600
# Print the result
print(sum_diff)
```

**Fig. 3.** R Code to classify by SVM Classifier

As it is demonstrated in the figure 3 we used a threshold 0 for classifying.

Parameters of SVM model are as in figure 4.

As you see accuracy is significantly higher than NN Classifier.(figure 5)

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.001
    epsilon:  0.1


Number of Support Vectors:  537
```

**Fig. 4.** parameters of SVM Classifier

```
"Accuracy: 0.795"
```

**Fig. 5.** Accuracy by SVM

```r
library(randomForest)
# Specify the predictor variables and the target variable
predictors <- names(train_data)[-which(names(train_data) == "Y")]
target <- "Y"
# Train the Random Forest model
rf_model <- randomForest(x = train_data[, predictors],
                         y = train_data[, target],
                         ntree = 100,  # number of trees in the forest
                         importance = TRUE)  # calculate variable importance
predictions <- predict(rf_model, newdata = test_data[, predictors])
# Assign 1 to positive predictions and -1 to negative predictions
predictions <- ifelse(predictions > 0, yes: 1, -1)
# Calculate the sum of differences and divide by 600
sum_diff <- sum(predictions - test_data$Y == 0) / 600
# Print the result
print(paste("Accuracy:", sum_diff))
```

**Fig. 6.** R Code to classify by RN Classifier

```
"Accuracy: 0.766666666666667"
```

**Fig. 7.** Accuracy by using RN Classifier

**Random Forest Classifier.** Now we apply a Random Forest Classifier by R code as in figure 6.

As you see similar to SVM, accuracy is significantly higher than NN Classifier but mostly less than using SVM classifier.(figure 7)

Note that number of trees does not affect the result significantly.

## Second part

In this part We train the data set on Head and also test on Head.(So here we train on 75 targets and 75 distractors and also test on 75 targets and 75 distractors)

```
successrate =

    0.5067
```

**Fig. 8.** accuracy while classifying by NNC

**Nearest Neighbor Classifier.** As you see we obtained accuracy 0.5067 by NNC.(figure 8)

**Support Vector Machine Classifier.** Parameters of SVM model are as in figure 9.

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.001
    epsilon:  0.1


Number of Support Vectors:  130
```

**Fig. 9.** parameters of SVM Classifier

```
"Accuracy: 0.52"
```

**Fig. 10.** Accuracy by SVM

As you see accuracy is higher than NN Classifier.(figure 10)

```
"Accuracy: 0.56"
```

**Fig. 11.** Accuracy by using RN Classifier

**Random Forest Classifier.** As you see similar to SVM, accuracy is significantly higher than NN Classifier and more than using SVM classifier.(figure 11)

## Third part

In this part We train the data set on Close body and also test on Close body.(So here we train on 75 targets and 75 distractors and also test on 75 targets and 75 distractors)

**Nearest Neighbor Classifier.** As you see we obtained accuracy 0.4645 by NNC.(figure 12)

```
successrate =

    0.5933
```

**Fig. 12.** accuracy while classifying by NNC

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.001
    epsilon:  0.1


Number of Support Vectors:  130
```

**Fig. 13.** parameters of SVM Classifier

```
"Accuracy: 0.509933774834437"
```

**Fig. 14.** Accuracy by SVM

**Support Vector Machine Classifier.** Parameters of SVM model are as in figure 13.

As you see accuracy is significantly less than NN Classifier.(figure 14)

**Random Forest Classifier.** As you see similar to SVM, accuracy is significantly less than NN Classifier and more than using SVM classifier.(figure 15)

```
"Accuracy: 0.536423841059603"
```

**Fig. 15.** Accuracy by using RN Classifier

## Forth part

In this part We train the data set on Far body and also test on Far body.(So here we train on 75 targets and 75 distractors and also test on 75 targets and 75 distractors)

```
successrate =

    0.4645
```

**Fig. 16.** accuracy while classifying by NNC

**Nearest Neighbor Classifier.** As you see we obtained accuracy 0.4645 by NNC.(figure 16)

**Support Vector Machine Classifier.** Parameters of SVM model are as in figure 17.

As you see accuracy is significantly higher than NN Classifier.(figure 18)

**Random Forest Classifier.** As you see similar to SVM, accuracy is significantly higher than NN Classifier but less than using SVM classifier.(figure 19)

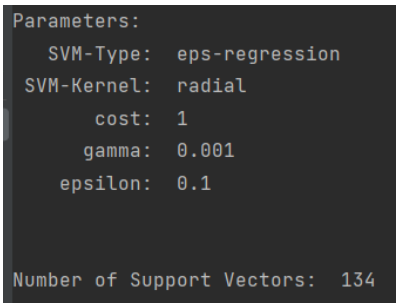


**Fig. 17.** parameters of SVM Classifier


**Fig. 18.** Accuracy by SVM


**Fig. 19.** Accuracy by using RN Classifier

## 5th part

In this part We train the data set on Medium body and also test on Medium body.(So here we train on 75 targets and 75 distractors and also test on 75 targets and 75 distractors)

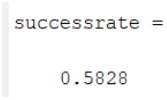


**Fig. 20.** accuracy while classifying by NNC

**Nearest Neighbor Classifier.** As you see we obtained accuracy 0.5828 by NNC.(figure 20)

**Support Vector Machine Classifier.** Parameters of SVM model are as in figure 21.

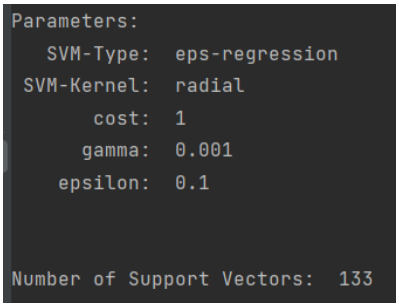


**Fig. 21.** parameters of SVM Classifier

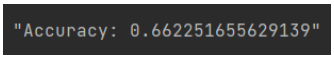


**Fig. 22.** Accuracy by SVM

As you see accuracy is higher than NN Classifier.(figure 22)

**Random Forest Classifier.** As you see similar to SVM, accuracy is significantly less than both NN Classifier and SVM classifier.(figure 23)


**Fig. 23.** Accuracy by using RN Classifier

## 6th part

Here we train the model on 300 targets and 300 distractors and test it on other 300 targets and 300 distractors.

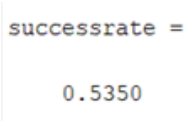


**Fig. 24.** accuracy while classifying by NNC

**Nearest Neighbor Classifier.** As you see we obtained accuracy 0.5350 by NNC.(figure 24)

**Support Vector Machine Classifier.** Parameters of SVM model are as in figure 25.

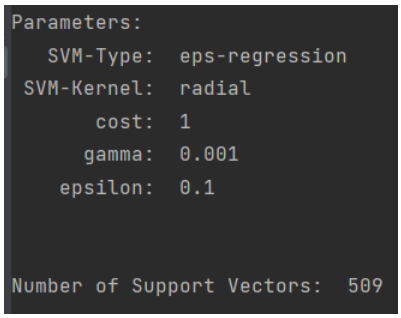


**Fig. 25.** parameters of SVM Classifier

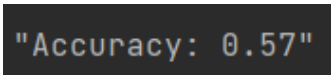


**Fig. 26.** Accuracy by SVM

As you see accuracy is higher than NN Classifier.(figure 26)

**Random Forest Classifier.** As you see similar to SVM, accuracy is significantly less than both NN Classifier and SVM classifier.(figure 27)

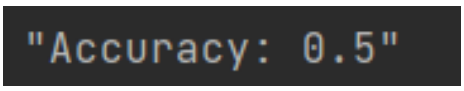


**Fig. 27.** Accuracy by using RN Classifier

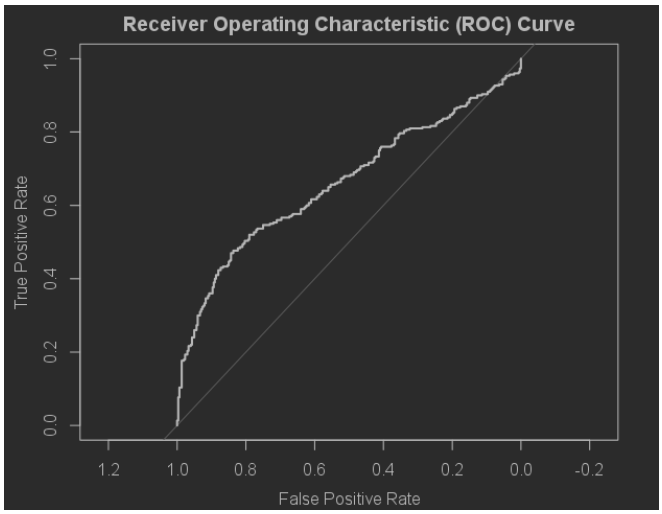


**Fig. 28.** ROC curve for all tested data

**ROC.** Figure 28 is ROC curve of the model tested on 600 test images.

Now we plot ROC for first 75 test target images which are close body(figure 29).

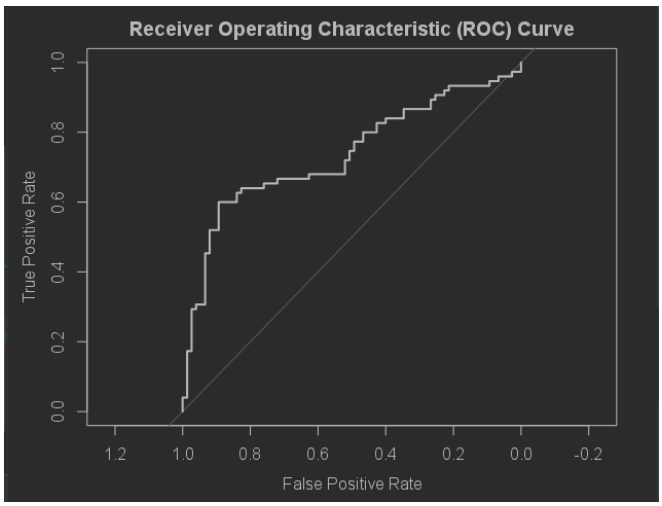


**Fig. 29.** ROC curve for close body data

Now we plot ROC for second 75 test target images which are far body(figure 30).

Now we plot ROC for second 75 test target images which are Head(figure 31).

Now we plot ROC for second 75 test target images which are Medium body(figure 32).

**d primes.** d' for these five datasets are as in figure 33.

We have plotted d' in figure 34. As you see it changes across datasets as same as the figure in article.

## 2. Conclusion

In the figures created in this homework we observed that mostly SVM classifier is the strongest classifier after learning procedure by Hmax model. We also calculated ROC for

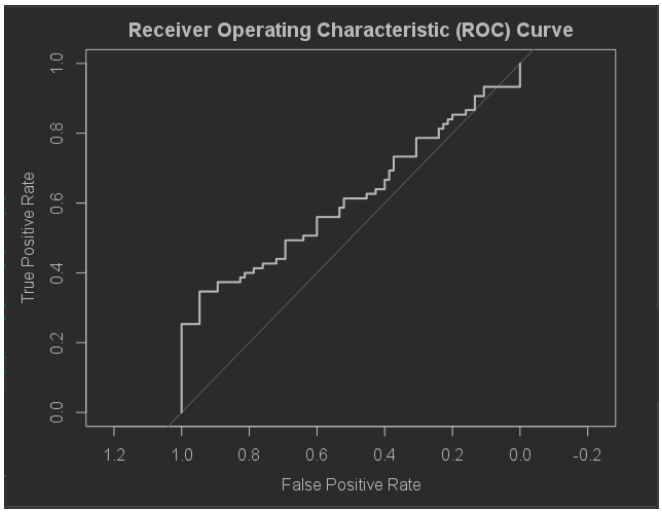


**Fig. 30.** ROC curve for far body data

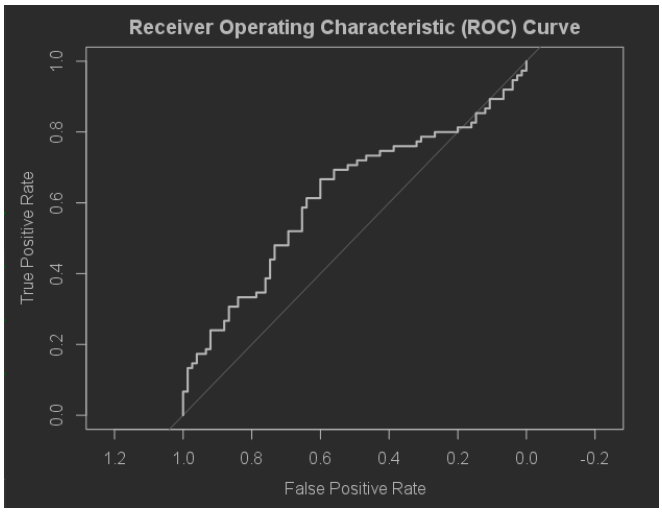


**Fig. 31.** ROC curve for Head data

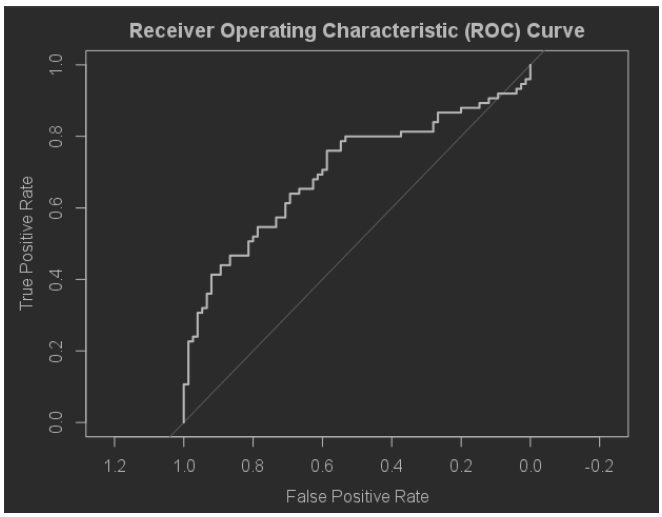


**Fig. 32.** ROC curve for Medium body data

```
d prime for all: 0.4674526
d prime for close body: 0.5211848
d prime for far body: 0.4330636
d prime for head: 0.4365834
d prime for medium: 0.4959176
```

**Fig. 33.** ROC curve for Medium body data

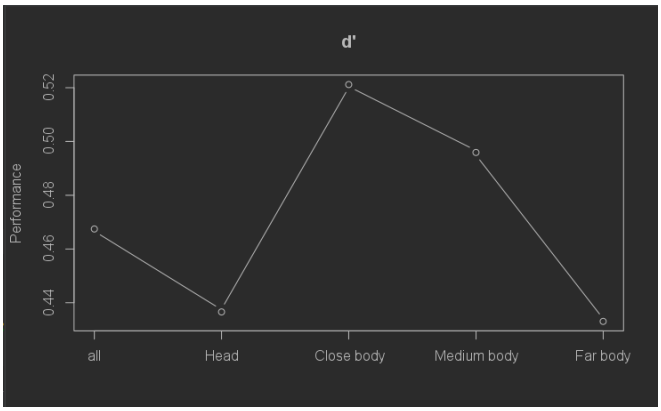


**Fig. 34.** d'

different views of body and showed that d' is higher for close body(as same as the article).

We also ran the model separately for each view and observed different classifiers performance while just one view have been in the learning procedure.

One phenomena of this homework was the first part of results that we obtained an accuracy which was approximately 20% higher than running the model on regular structure of dataset. In this part we trained the data set on Head and Medium body and tested on Far and Close body.

Further more also could run the model and investigate its accuracy on noisy images. We use salt and pepper noise or gaussian noise.

## References

1. A feedforward architecture accounts for rapid categorization; Thomas Serre, Aude Oliva, and Tomaso Poggio

2.Hubel DH, Wiesel TN (1968) J Phys 195:215243.

3. Perrett D, Oram M (1993) Image Vision Comput 11:317333.

4. Kobatake E, Tanaka K (1994) J Neurophysiol 71:856867.

5. Tanaka K (1996) Annu Rev Neurosci 19:109139.