# 1 Hyperparameters optimization

## 1.1 K-nearest neighbors

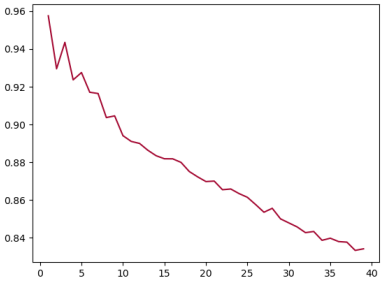


Figure 1

For K-nearest neighbors, we only need to optimize $k$, aka the number of neighbors used to predict the label of an input. We used cross-validation to optimize $k$. Figure 1 shows the validation accuracy as a function of $k$ where, we clearly see that knn perform best with a small $k$. Choosing a small k leads, in general, to some amount of overfitting. It does indeed overfit in this case (figure 1 compared with our results next page) but we still got the best performance on the test dataset with $k = 1$.

## 1.2 Deep Network

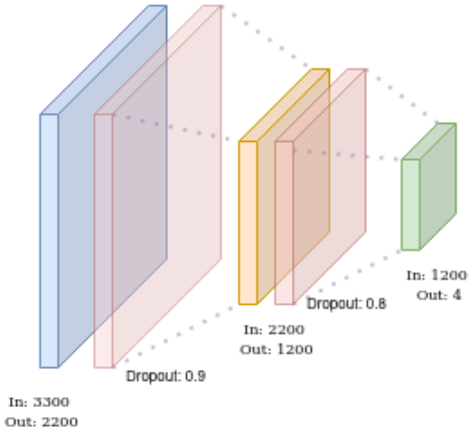


Figure 2

There are a lot of parameters to optimize with deep networks (learning rate, n of hidden layers, n of nodes, ...). The main problem with h36m data was to avoid overfitting. Indeed, due to the low ratio $N/D$ of the data, the deep model was naturally prone to overfitting.

Figure 2 shows the architecture that gave us the best accuracy. It uses two hidden layers respectively of 2200 and 1200 neurons. With this many neurons, we had to have a lot of regularization in order to avoid overfitting. Therefore, we added two dropout layers respectively of 90% and 80%. For the same reason, we also added ridge regularization with the $weight\_decay = 1e^{-5}$ in PyTorch.

We used the Adam optimizer for this problem, since it gave us a better result than standard gradient descent with manual learning-rate decay.

Choosing the starting learning-rate was done by trial and error, since it depended a lot on the actual architecture of the neural network. We ended-up using a learning rate of $1e^{-5}$ which wasn't the fastest to converge but gave us a consistent result.

## 1.3 PCA

### 1.3.1 Linear regression

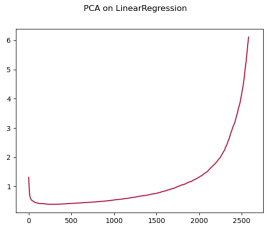


Figure 3

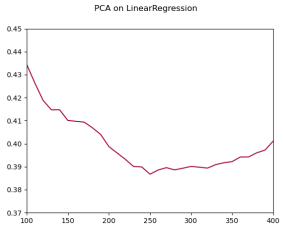


Figure 4

In milestone one, linear regression performed poorly due to the high dimensionality of h36m data. It comes as no surprise that pca helped tremendously the linear regression model by limiting the size of the input. Figure 3 and 4 shows the loss as a function of the dimension of the data after pca reduction. Linear regression performs best with $d = 250$ landing in the sweet spot between too much or too little information.

### 1.3.2 Ridge regression

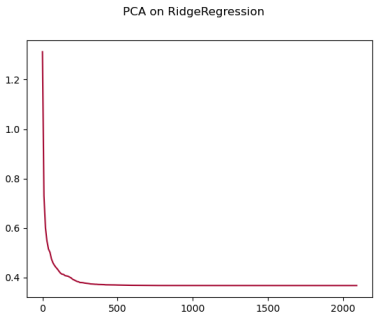


Figure 5

ridge regression already performed well with $\lambda = 170$, which resulted in a loss of 0.367. Pca at best matches this result (see Figure 5). Since the computation of ridge regression isn't that significant, pca isn't useful.

### 1.3.3 Logistic Regression / KNN

Logistic regression and KNN didn't improve with pca, but reducing dimensions had no impact on the accu-

racy (see Figure 7 and 6). We can therefore use pca to slightly reduce the computation time of our model.

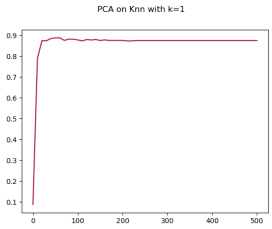


Figure 6


Figure 7

#### 1.3.4 Deep Network

Complex neural networks takes a lot of time to train, so pca is a great candidate to reduce computing time. We indeed got satisfying accuracy using pca in much less time. Moreover, our model tended to overfit less with pca. However, to have the maximum result we had to refrain from using pca since reducing dimensions costed us about 2 - 3 % accuracy.

# 2 Results and discussion

| linear w/ PCA | Ridge | Logistic | Knn |
|---|---|---|---|
| $d = 250$ | $\lambda = 170$ | lr = 1e−4<br>iter = 10 | $k = 1$ |
| MSE = 0.387 | MSE = 0.367 | Acc = 78.94 %<br>F1 = 0.710 | Acc = 89.8 %<br>F1 = 0.875 |

Figure 8

| Deep Network | | |
|---|---|---|
| iter = 150 | lr = $1e^{-5}$ | weight-decay = $1e^{-5}$ |
| F1 = 0.953 | Acc = 96.1 % | |

Figure 9

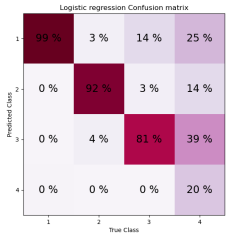


Figure 10

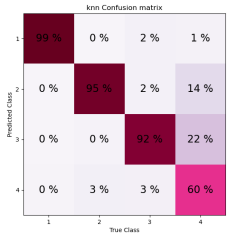


Figure 11

## 2.1 K-nearest neighbors

Figure 11 is the confusion matrix of knn. We can see that, unlike logistic regression in figure 10, it can somewhat recognize class 4 data. This make knn a more robust algorithm than logistic regression for this classification problem.

## 2.2 Deep Network

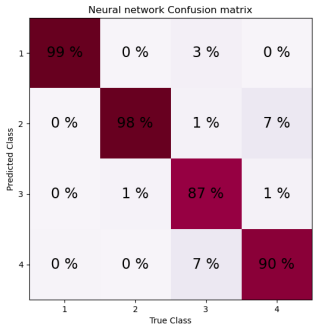


Figure 12

The neural network is by far the better algorithm in all metrics. Figure 12 shows how well the neural network can differentiate the 4 classes. Despite being computationally more expensive than knn the neural network performs significantly better scoring 6% higher in accuracy

## 2.3 Conclusion

| class | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| n-labels | 809 | 770 | 550 | 460 |

Figure 13: training labels

| class | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| n-labels | 131 | 222 | 163 | 102 |

Figure 14: test labels

We have tested multiple algorithms / architecture on the same regression / classification problem.

On regression problem, while using pca with linear regression gave us a similar result to ridge regression, the loss was slightly higher. Ridge regression is the clear winner here, having a lower loss and a smaller execute time.

On classification problem, it gets more interesting. While The neural networks clearly outperforms the two other competitors, it comes at a cost. Indeed, the training of the deep network is much longer than the others.

Furthermore, we can see in figure 13 and 14 that the distribution of classes is not the same. If we had more class 4 data in the training dataset, knn could maybe have a closer result to the deep network while still being a computationally cheaper alternative.