# Hyperparameter optimization by supervised learning for image recognition

Group 12
Caroline Rippey, Alexandre Dutour, Simon Bayle

POSTECH

# Table of contents

# Optimization of machine learning algorithms

One could want to optimize an algorithm in order to be :

- Time efficient
- Cost efficient

Optimizing a machine learning algorithm consists of optimizing a **black box** problem. One has multiple ways to optimize it :

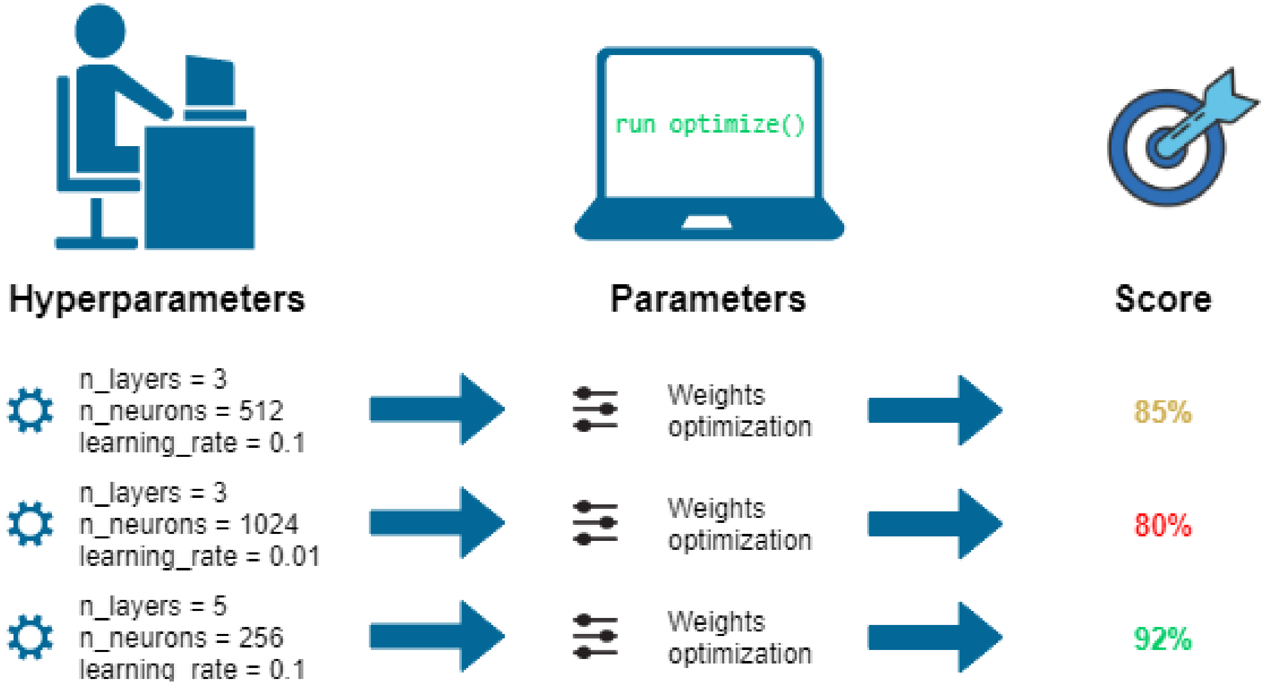
- By simply **increasing the computational power**, we will then improve the execution time at the expense of the costs
- Optimizing the algorithm **hyperparameters**

# Hyperparameters of a ML algorithm

A **hyperparameter** is a parameter that is set before the beginning of the learning processes and impacts the effectiveness of a model training.

This can be :

- The learning rate
- Properties of the neural network (number of layers and neurons)
- Batch size
- Number of epochs
- …



# Supervision method

We chose to use **supervised learning** in order to perform our hyperparameters optimization.

**Supervised learning** is defined by :

- A **labeled dataset**. All the inputs and outputs of the dataset is correctly labeled
- Thanks to this, the model can measure precisely :
  - His accuracy over time
  - His loss over time

Like this, it is easier to classify accurately how hyperparameters sets perform

A harder but more flexible method would have been **unsupervised learning**

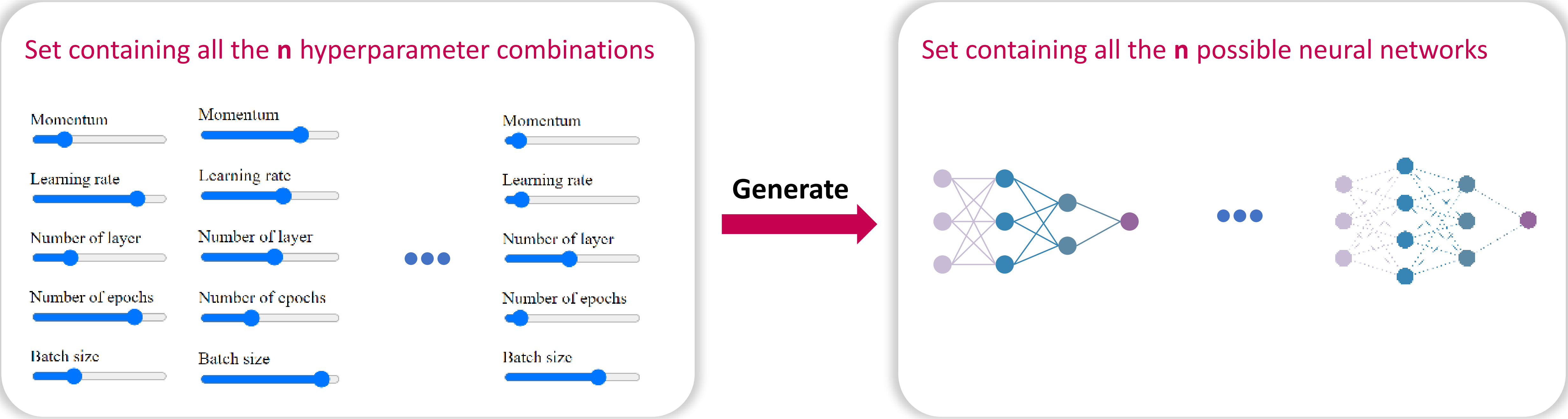
# Optimization of a hyperparameter

This can be achieved by using multiple techniques such as :

- Grid search
- Random search
- **Bayesian optimization**

# Grid search

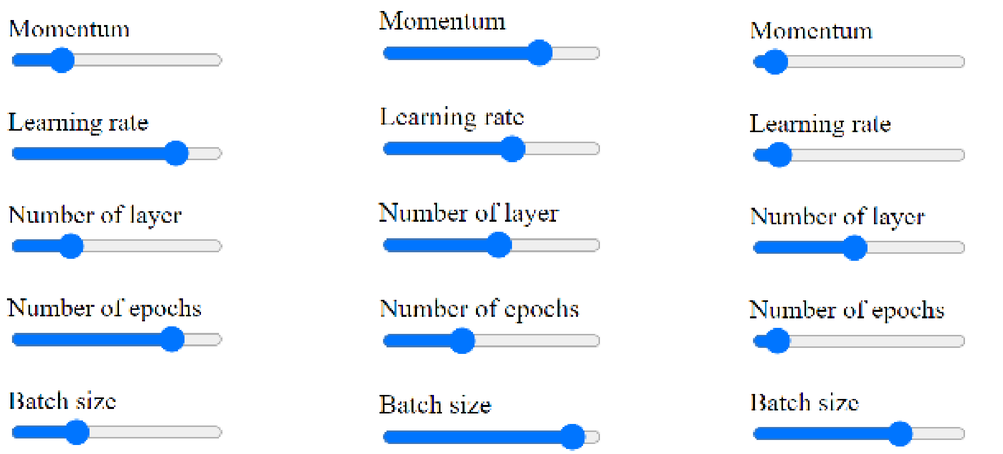
This algorithm will generate **all** the neural networks possible based on **all** the hyperparameters possible combinations and train all of them to keep the best results



# Random grid search

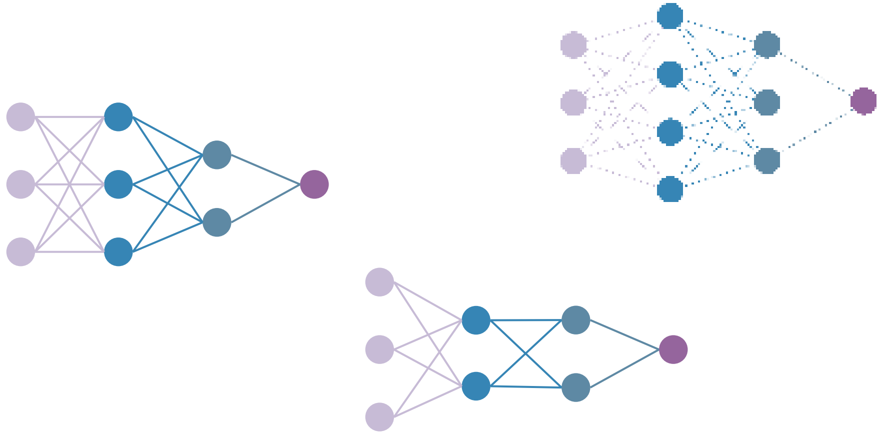
This algorithm will generate **x out of n** neural networks possible based on **x out of n** of the hyperparameters possible combinations and train all of them to keep the best results

Set containing **x out of n** hyperparameter combinations




**Generate**

Set containing **x out of n** possible neural networks



# Introduction to Bayesian Optimization

Like classical optimization : $Min\ f(x)\ , x\ \in\ X$

Bayesian optimization constructs a probabilistic model for $f(x)$ and then exploits this model to make decisions about where in $X$ to next evaluate the function, while integrating out uncertainty.

Use all information of the function.

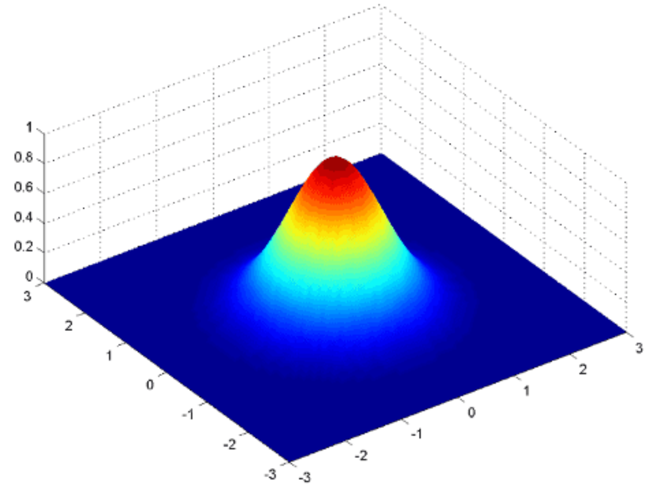
The 2 choices of Bayesian Optimization :

- Select a prior over functions that will express assumptions about the function being optimized (Gaussian)
- Choose an acquisition function that will determine the next point to evaluate.

# Bayesian Optimization Gaussian Process

Gaussian Processes (GP) :

**Property** : Assume that the GP is a function define by $f : X \rightarrow R$. Any finite set of N points $\{x_n \in X\}_{n=1}^{N}$ induces a multivariate Gaussian distribution on $R^N$ . (Gaussian generalized to N dimensional space).




$x \in X : \; x_{next} = argmax_x a(x), \; a(x\,;\{x_n, y_n\}, \theta) : X \rightarrow R^+$

Mean and Variance : $\mu(x\,;\{x_n, y_n\}, \theta), \; \sigma^2(x\,;\{x_n, y_n\}, \theta)$

$x_{best} = argmin_{x_n} f(x_n)$

The cumulative distribution function : $\Phi(x)$

# Bayesian Optimization method used in the project

- There are different types of improvements for the Bayesian steps :

  - One intuitive strategy is to maximize the probability of improving over the best current value. Under the GP this can be computed analytically as :

$$a_{PI}(x\,;\,\{x_n,y_n\},\theta)=\Phi(\gamma(x)),\quad \gamma(x)=\frac{f(x_{best})-\mu(x\,;\,\{x_n,y_n\},\theta)}{\sigma(x\,;\,\{x_n,y_n\},\theta)}$$

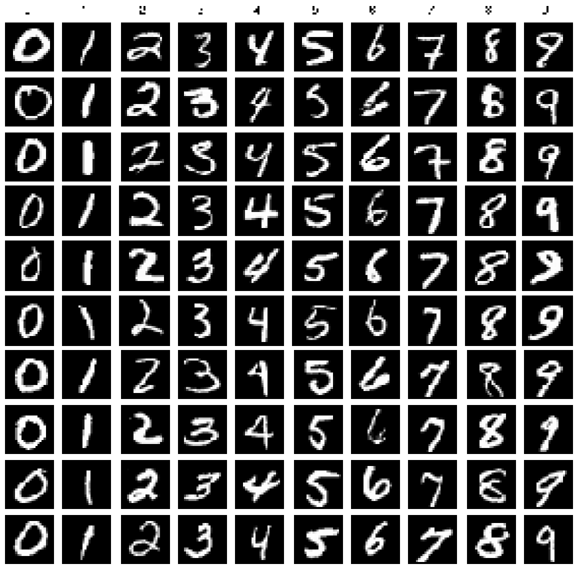
  - Alternatively, one could choose to maximize the expected improvement (EI) over the current best value. This also has closed form under the Gaussian process:

$$a_{EI}(x\,;\,\{x_n,y_n\},\theta)=\sigma(x\,;\,\{x_n,y_n\},\theta)(\gamma(x)\,\Phi(\gamma(x))\;+\;N\,(\gamma(x)\,;\,0,1))$$

# Data Processing

Data Set for Image Classification:

MNIST Handwritten Data



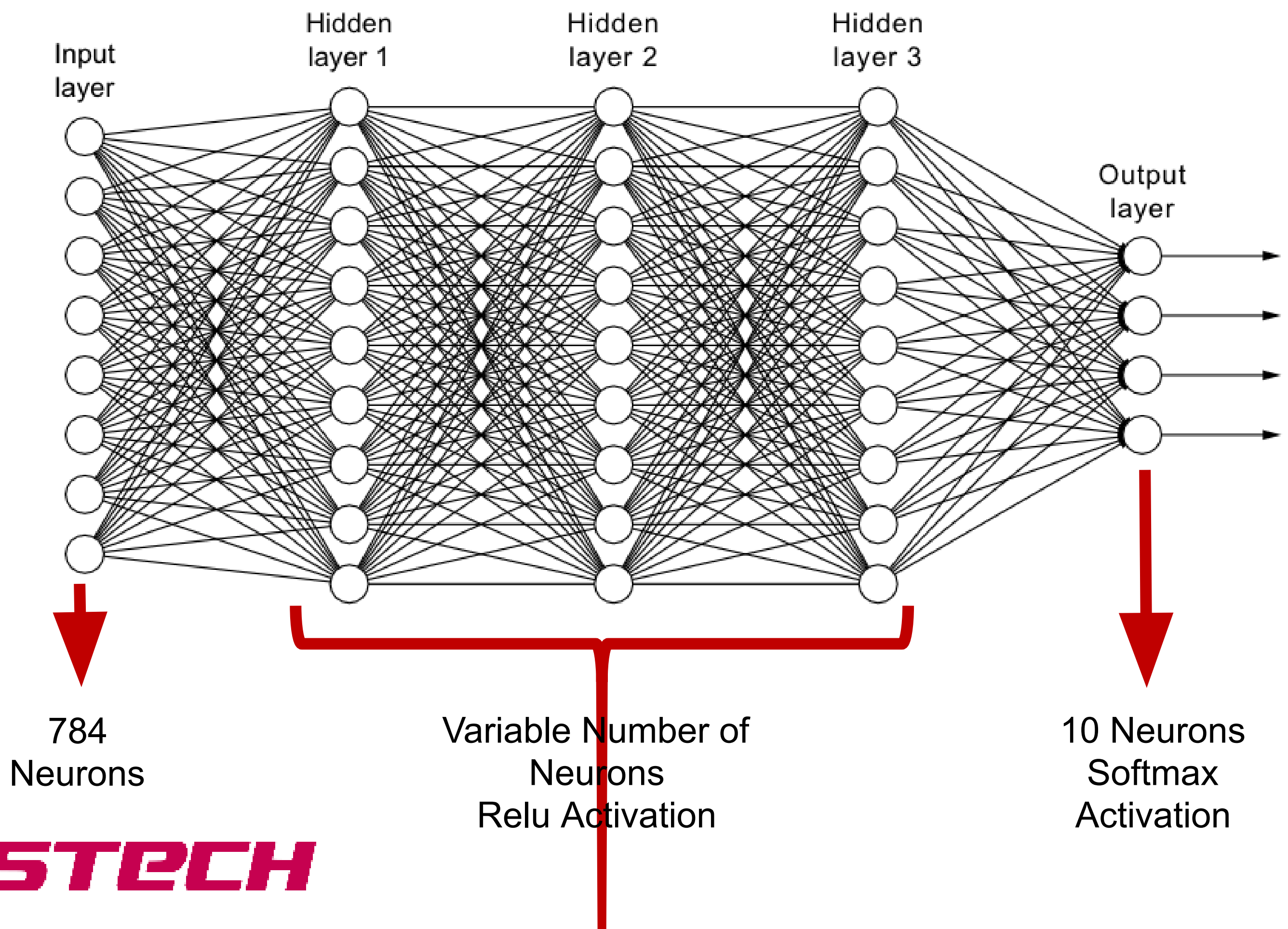
https://www.researchgate.net/figure/Example-images-from-the-MNIST-dataset_fig1_306056875

Data Processing:

- Splitting into training and testing data
- Flattening Data
- Rescaling pixel values

# Neural Network Construction

Dense Neural Network




Variable inputs of the neural network construction for hyperparameter testing:

- Learning rate
- Momentum
- Number of neurons per Layer

Constant hyperparameters:

- Number of layers
- Batch Size
- Epochs

POSTECH

# Hyperparameter Search Space

3 tested hyperparameters within a given search space bound

Learning Rate:
[0.0001, 1]

Momentum:
[0,1]

Number of Neurons per hidden Layer:
{16 ,32, 64}

Constant Hyperparameters

Number of Layers:
3

Epochs:
4

Batch Size:
32

# Type of Hyperparameter Optimization

Applied 3 types of hyperparameter optimization:

- Grid
- Random
  - Different distributions
    - Continuous
    - Grid
- Bayesian
  - Different distributions
    - Continuous
    - Grid
  - Different Acquisition Functions
    - Probability of Improvement
    - Expected Improvement

Compare using maximum achieved accuracy over number of iterations

# Experiments

We did multiple experiments in order to see the impact of hyperparameters on neural networks results.

- We tried to optimize hyperparameters using 3 techniques :
  - Grid search
  - Random grid search
  - Bayesian optimization

  And compare how the new parameterized neural network performed with these hyperparameters.

- Separately, we also tried to see the impact of the number of epochs on the final performance

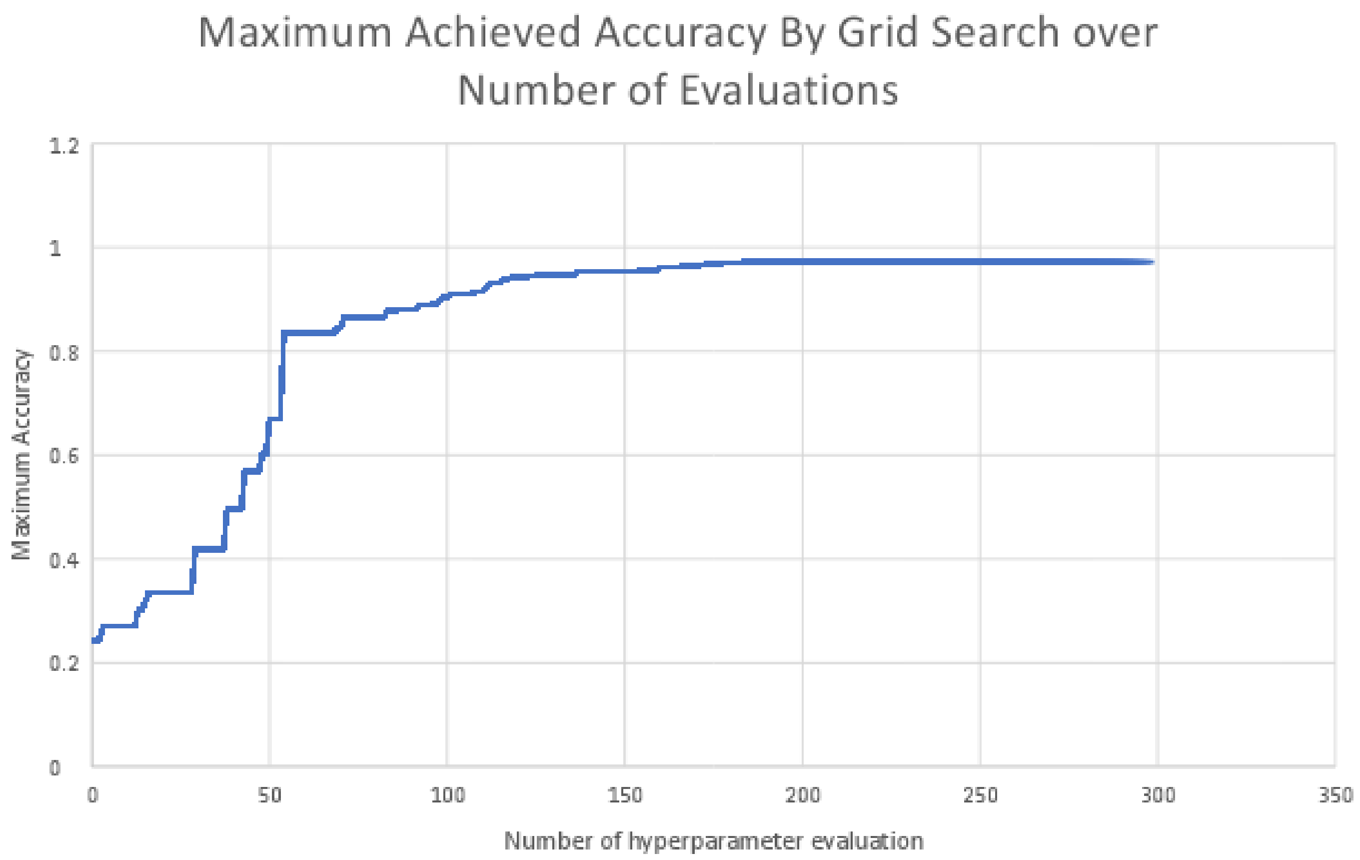
# Grid search

Setup:

- Search space : all the following combinations of :
  - **Learning rate** : $10^{-n}$ with integer $n\ in\ [-4, 0]$
  - **Momentum** : $[0, 1]$ with a step size of $0.1$
  - **Number of hidden layers** : 3
  - **Number of nodes per layer** : $[x, y, z, 10]$ with $x, y, z\ in\ \{16,32,64\}$

# Grid search

Result:



# Random search

Grid Setup:

- Search space was random combinations of :
  - **Learning rate** : $10^{-n}$ with integer $n \ in \ [-4, 0]$
  - **Momentum** : $[0, 1]$ with a step size of $0.1$
  - **Number of hidden layers** : 3
  - **Number of nodes per layer** : $[x, y, z, 10]$ with $x, y, z \ in \ \{16{,}32{,}64\}$

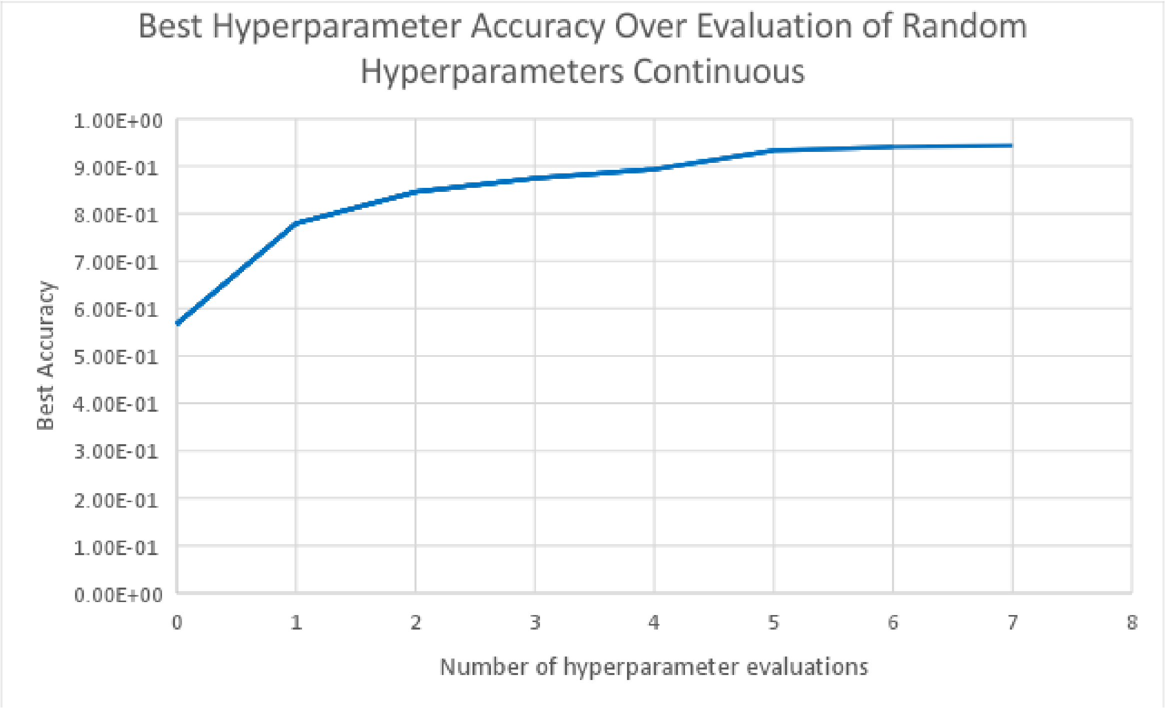
Continuous Setup:

- Search space : random combinations of :
  - **Learning rate** : $10^{-n}$ with $n$ from uniform distribution $[-4, 0]$
  - **Momentum** : from uniform distribution $[0, 1]$
  - **Number of hidden layers** : 3
  - **Number of nodes per layer** : $[x, y, z, 10]$ with $x, y, z \ in \ \{16{,}32{,}64\}$

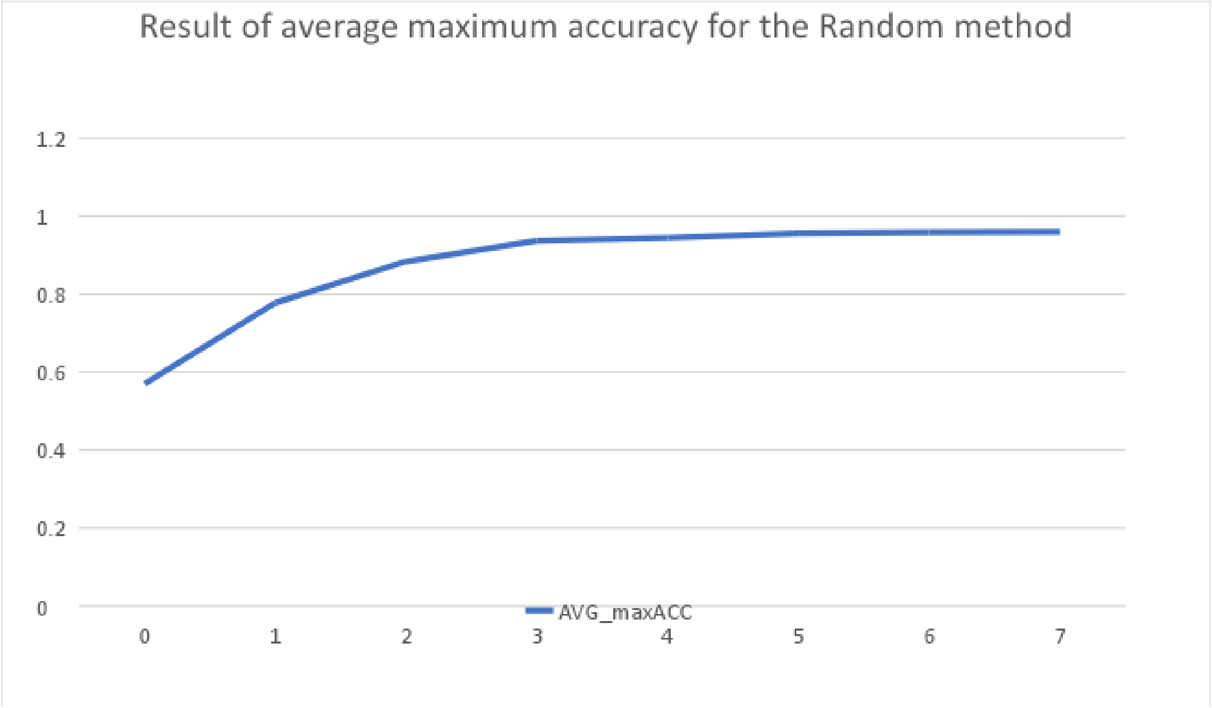
# Random search

Result:

Continuous




Grid



# Bayesian optimization

Grid Setup:

- Search space : random combinations of :
  - **Learning rate** : $10^{-n}$ with integer $n\ in\ [-4,0]$
  - **Momentum** : $[0,1]$ with a stepsize of $0.1$
  - **Number of hidden layers** : 3
  - **Number of nodes per layer** : $[x,y,z,10]$ with $x,y,z\ in\ \{16{,}32{,}64\}$

Continuous Setup:

- Search space : random combinations of :
  - **Learning rate** : $10^{-n}$ with $n$ from uniform distribution $[-4,0]$
  - **Momentum** : from uniform distribution $[0,1]$
  - **Number of hidden layers** : 3
  - **Number of nodes per layer** : $[x,y,z,10]$ with $x,y,z\ in\ \{16{,}32{,}64\}$

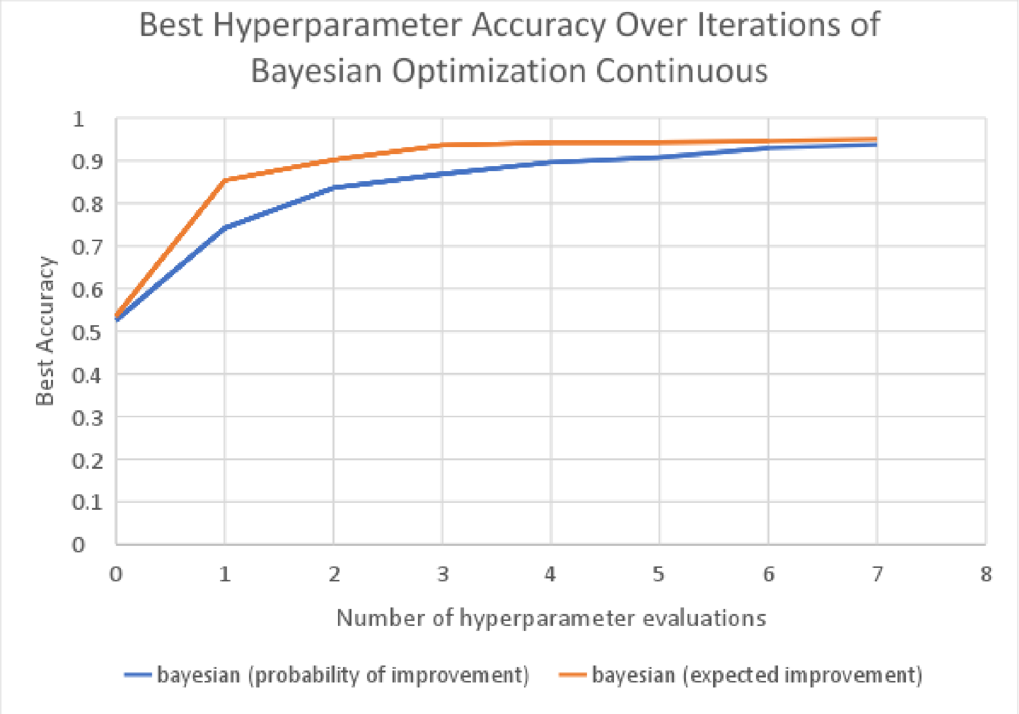
For each setup:

- Probability of Improvement Acquisition Function
- Expected Improvement Acquisition Function

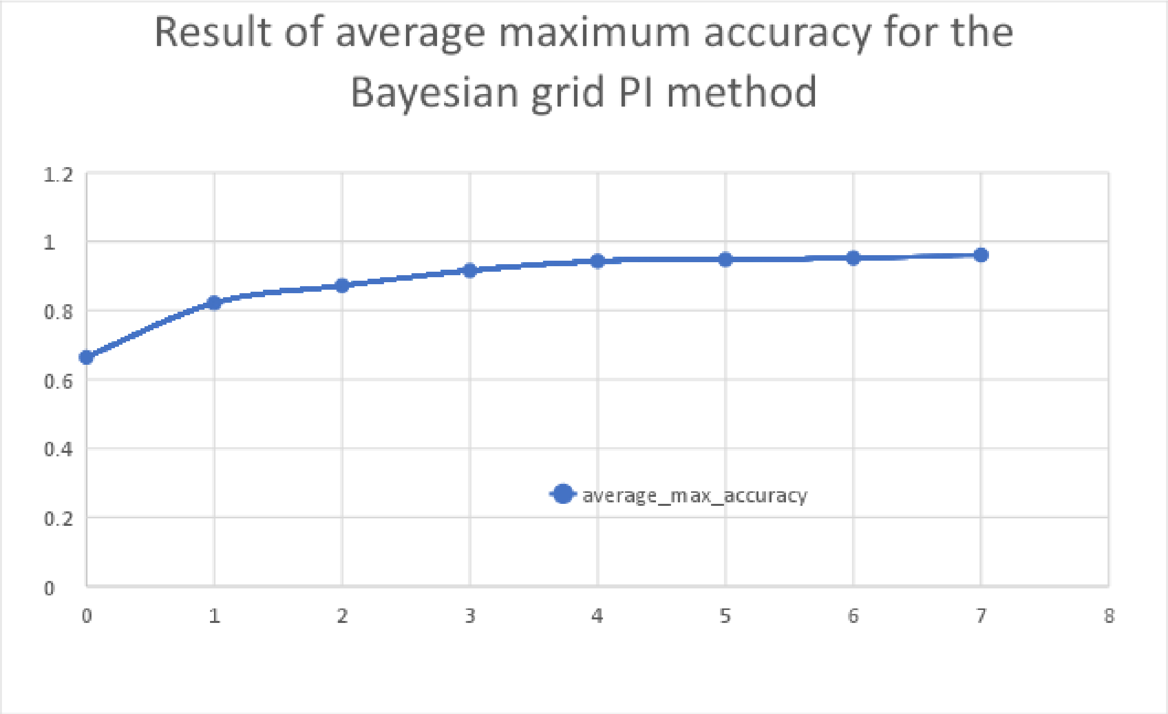
# Bayesian optimization

Result:

Continuous




Grid

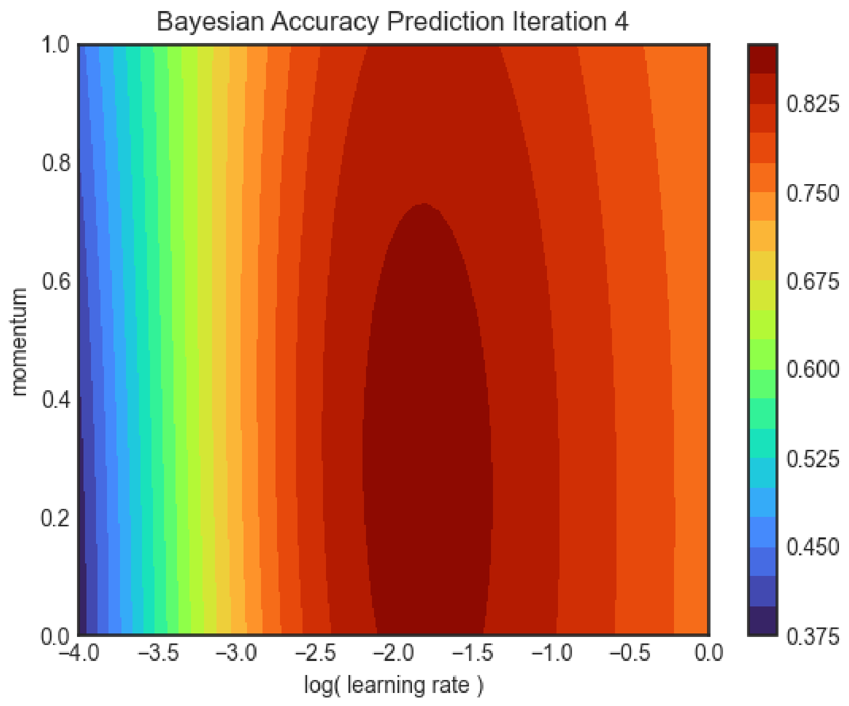


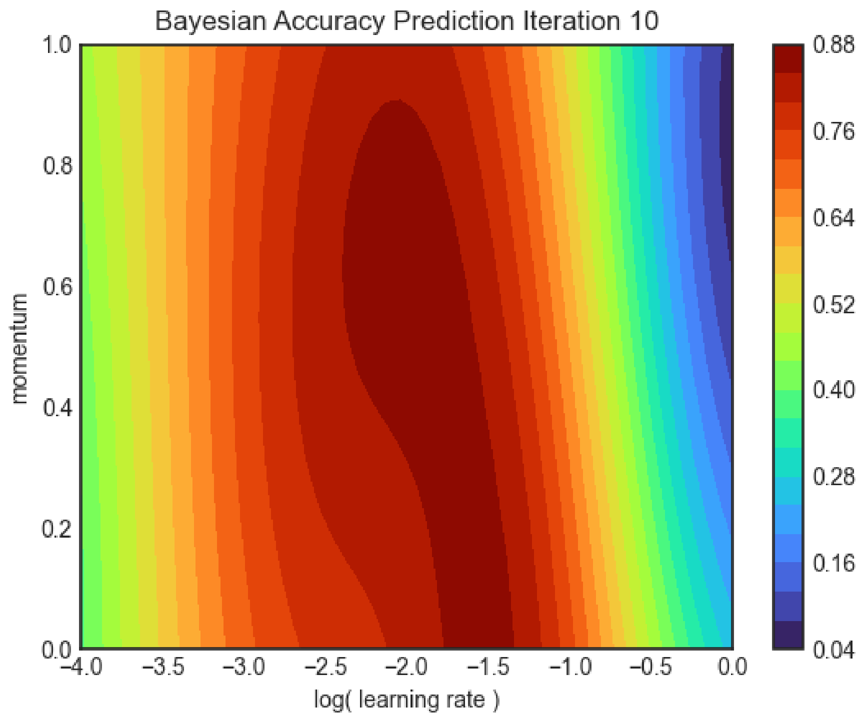
# Visualizing Bayesian Learning

Setup:
- Search space: Continuous hyperparameter search space
- Acquisition function: Expected improvement acquisition function
- Number of neurons per layer: fixed to [32,64,16,10]
- Optimized: Learning rate and momentum

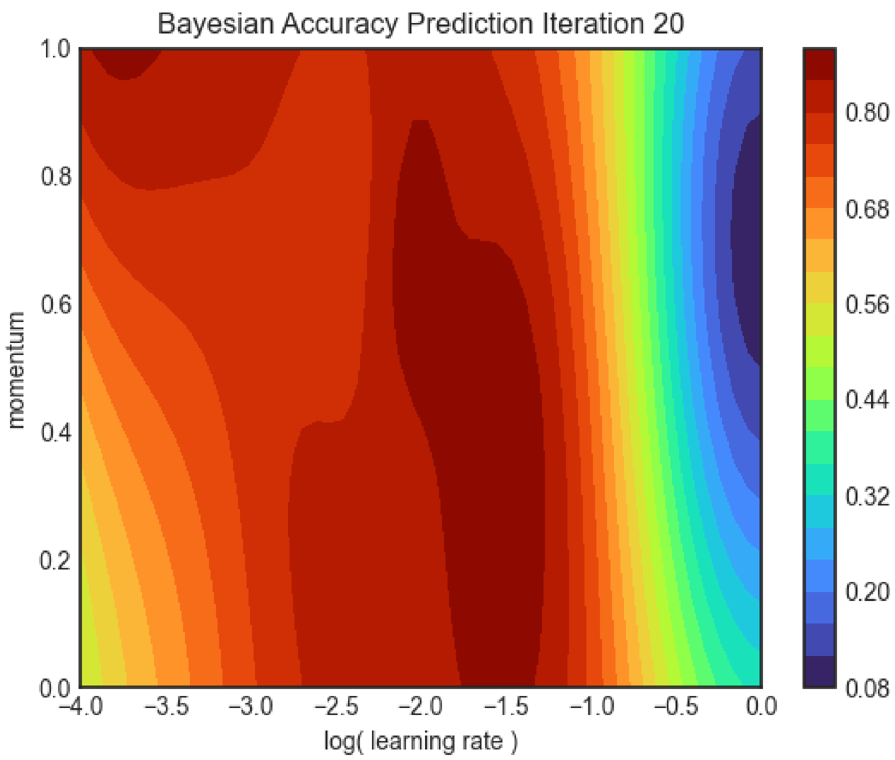
Result
- Heat map plot of Bayesian predicted accuracy of setting of learning rate and momentum
- Visualize predictions after a different number of iteration of the Bayesian algorithmc

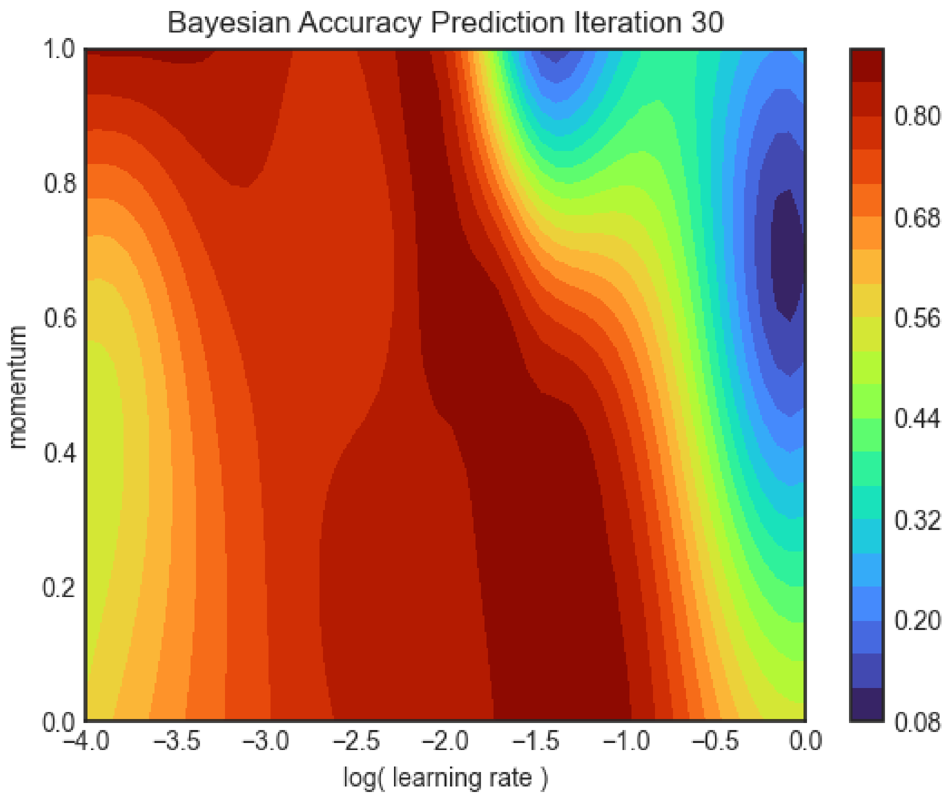
# Visualizing Bayesian Learning



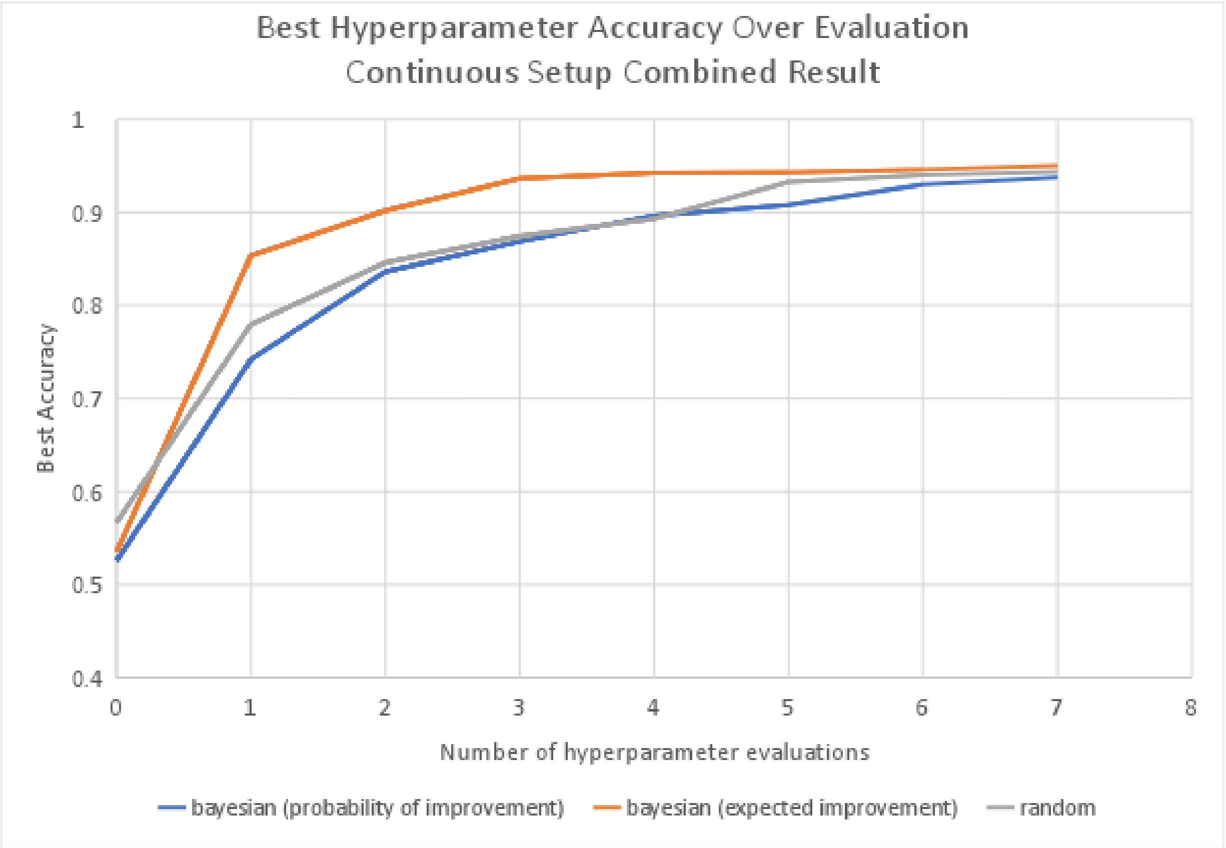



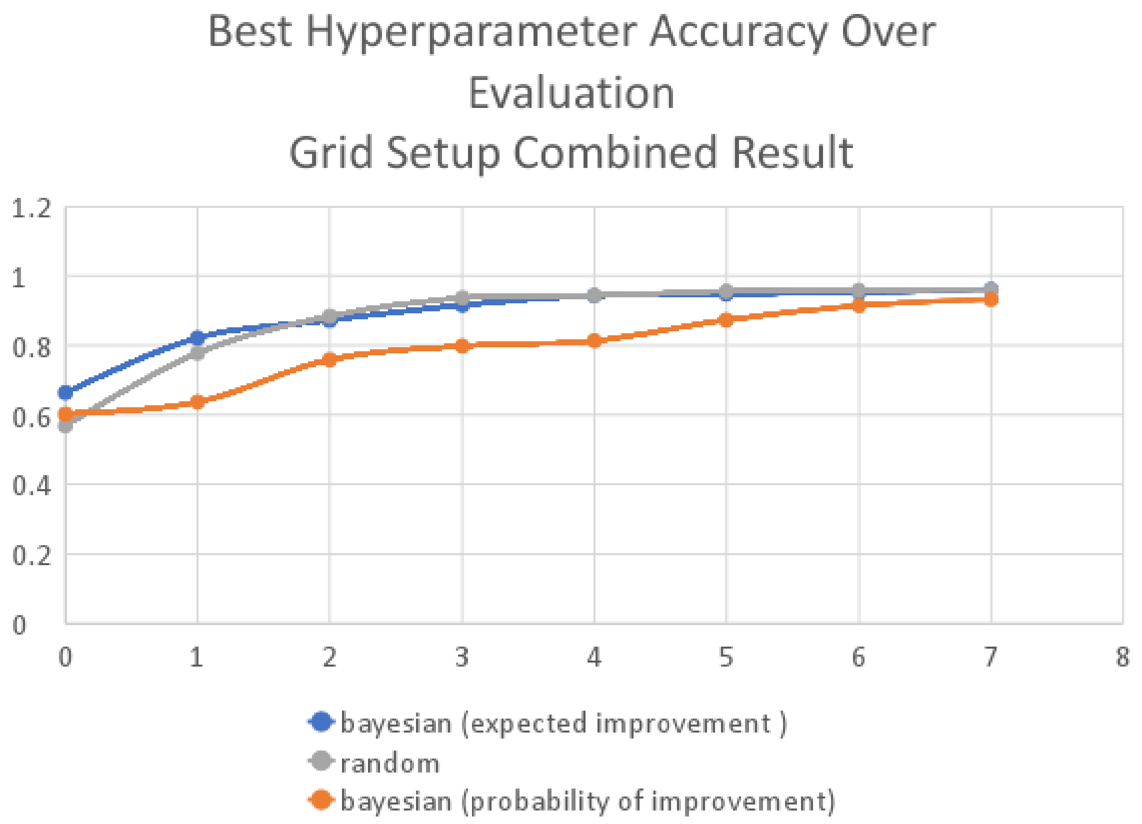
# Visualizing Bayesian Learning



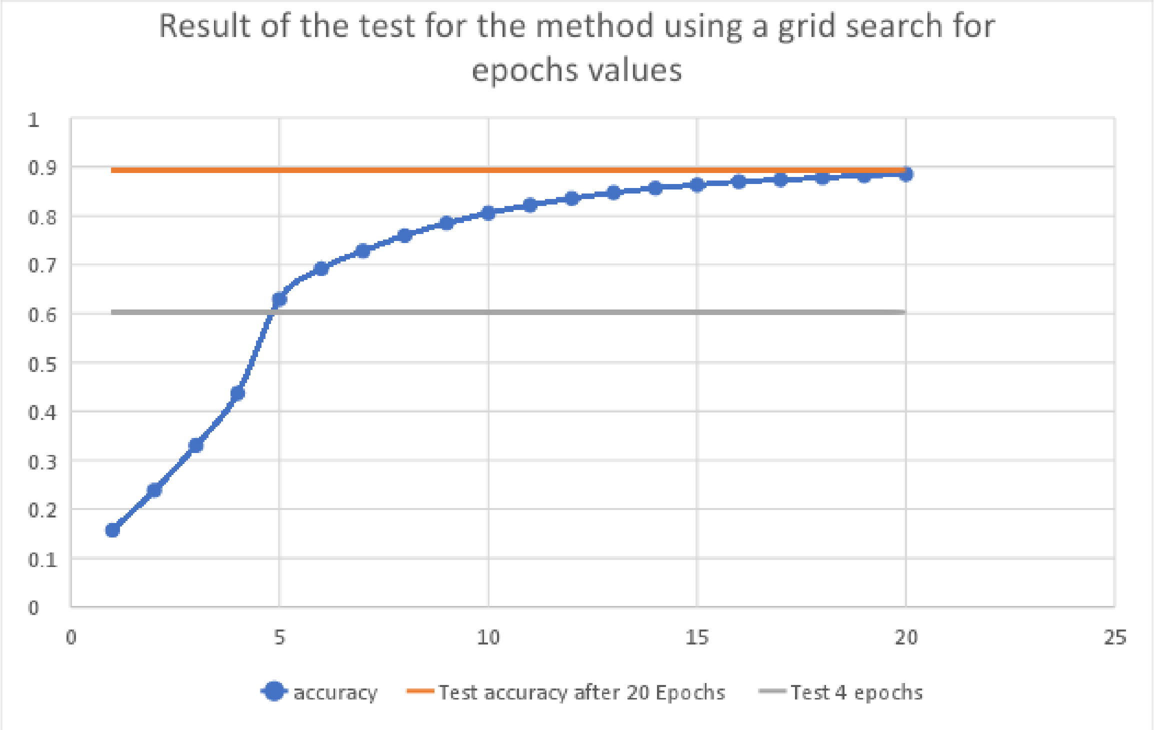
# Corelated results

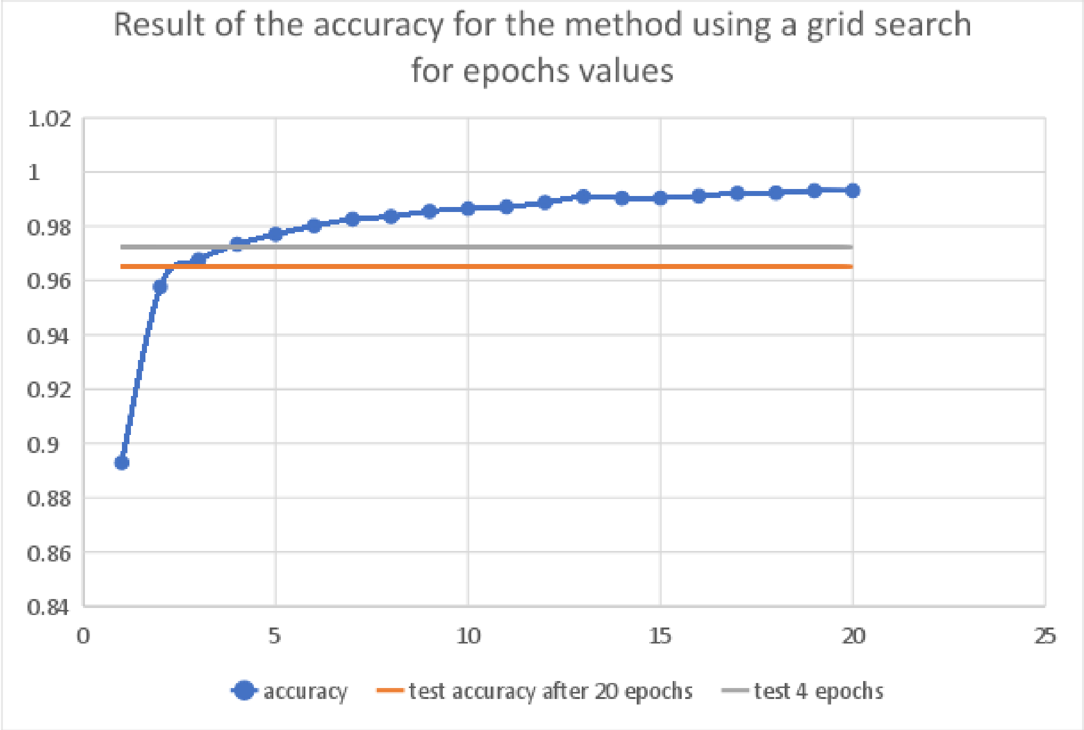


# Impact of epochs

Setup:



# Conclusion

Observation:

- Neural network accuracy varied from ~0.1 to ~0.97 after 4 epochs based on various hyperparameter settings

Potential Reasoning:

- Hyperparameter have significant effects on how the SGD optimization algorithm functions and whether it converges to an optimal point

Conclusion:

- Finding good hyperparameter settings is important for efficient training and good accuracy of a neural network

# Conclusion

Observation:

- Grid search took takes massive number of evaluations to complete while random search and Bayesian optimization will find a good set of hyperparameter in very few evaluation

Potential Reasoning:

- There are many points in the search space that are good sets of hyperparameter that even by random sampling you are likely to discover a good set of hyperparameter in very little time
- Bayesian optimization provide allows sampling that is more likely to be a good set

Conclusion:

- It is best to use either random search or Bayesian optimization for hyperparameter tuning

# Conclusion

Observation:

- Heat plots show that accuracy predictions vary greatly across different learning rates while moment has less effect on accuracy predictions

Potential Reasoning:

- Learning rate can affect significantly how fast a model converges or whether is converges at all
- Learning rate can determine whether a model gets stuck in a local minimum
- Momentum has less effect on convergence behavior

Conclusion:

- It might be important to optimize learning rate than momentum

# Conclusion

Observation:
- Bayesian optimization results were not significantly better than random search results

Potential Reasoning:
- The search space was small and many candidates in the search space were equally good
- Convergence to optimal hyperparameters took very few iteration not allowing Bayesian optimization to develop a good model

Conclusion:
- Bayesian optimization may be better applied to more complex hyperparameter setting with larger search spaces

# Conclusion

- Finding a good set of hyperparameter is an important part of creating an efficient functioning neural network
- Hyperparameter setting have significant effects on convergence behavior of a neural network
- Some hyperparameters have more effect on convergence rate than others
- Both random search and Bayesian optimization can both be useful methods to tune hyperparameters which is a better fit may be dependent on the search space complexity