University of Nottingham
UK | CHINA | MALAYSIA

Predicting Binding Affinity with Convolutional Neural Networks

Submitted April 2022, in partial fulfillment of the conditions
for the award of the degree **BSc Computer Science with Artificial Intelligence.**

Chenkai HU
20123721

Supervised by Prof. Colin Johnson

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Abstract

Binding affinity is a crucial measurement in drug discovery. Prediction of binding affinity is therefore a widely attempted problem. With the recent development of machine learning, deep learning methods (like Convolutional Neural Network) are applied with manually extracted features. In this project, a new form of input data (Molecular Interaction Field) is used in training a model based on CNN. Different architectures and training settings are implemented and experimented to tune the model. Optimal model found could achieve a R^2 of -0.1 with a MAE of 0.07.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr.Ferrante Neri and Prof.Colin Johnson for their support and guidance. I would also like to express my sincere gratitude to Prof.Jonathan Hirst from chemistry major for providing chemistry background knowledge, guidance in processing the data and the access to high performance computing. Without him, the large portions of the project would not have been possible. Additionally, I would like to thank David Rogers for always leading an ear whenever I was stuck. Moreover, I would like to thank my fellow Chuxiong WU who shared his disk quota on HPC with me and gave support when I implemented the ResNet architecture. Finally, I would like to thank my parents and my girlfriend for their unwavering belief in my ability to succeed.

Contents

Chapter 1

Introduction

1.1 Motivation

Machine learning is a sub-topic of artificial intelligence in the spotlight. It is famous for being able to imitate intelligent human behavior without being explicitly programmed to do so.In particular, machine learning shows great potential of exploring the information hidden in massive figures. Applying this technique to real world problem is a trend that could be found in different disciplines ranging from fiance (fraud detection [1], option pricing [2]) to engineering. Chemistry, for instance, is such an area that have a growing trend of utilizing statistical machine learning techniques [3].

With improvements in quantum-mechanical approximations and growing capability of electronic-structure codes, time is ripe for machine learning methods to provide insights in this area's research [4]. The application of machine learning technique is promising as it is possible to reduce the cost and time spent on assays.[5] Powered by statistical modelling, the improvement in accuracy and efficiency of molecular simulations has been established [4]. This statistical approach could also shed light on new perspectives for research (for example, in amorphous materials study [6]) and design of new chemical compound [7].

Deep learning, as a subset of machine learning, gained attention in research area for its capability of accurately modelling and making predictions on problems with unstructured data. Predicting binding affinity is such a problem in chemistry to facilitate drug discovery, and yet deep learning plays more and more an important role in solving it.

1.2 Background

1.2.1 Problem

Quantitative Structure-Activity Relationship (QSAR) is one of the useful quantitative techniques to filter candidate compound with undesirable characteristics and help with drug development. In this project, deep learning techniques are applied to the prediction of binding affinity. Binding affinity is a metric in QSAR of strength of the binding interaction between a single biomolecule (e.g. protein or DNA) to its ligand/binding partner (e.g. drug or inhibitor), helpful in indicating the ligand's efficiency. A popular measurement of binding affinity is Half-maximal inhibitory concentration (IC_{50}), which represents how much of a particular inhibitory substance is needed to inhibit a biological process by 50%. It is more intuitive to compare using the negative log of IC_{50}, referred to as pIC_{50}, as larger the pIC_{50} value indicates a smaller concentration needed for the inhibiting process.

1.2.2 Neural network

Artificial Neural Networks (ANN) is the foundation of modern deep learning. In this project, a derivative of ANN (CNN) is to be used, it is therefore worthwhile to bring up some fundamental concepts for later references.

ANN get inspiration from the idea of mimicking human's nervous system and preserves a lot terminology from neuroscience (1.1). The basic unit component of neural network is called **neurons**, which receives signals, processes it and transmit signal to connected neurons based on an non-linear **activation function** of the sum of inputs.

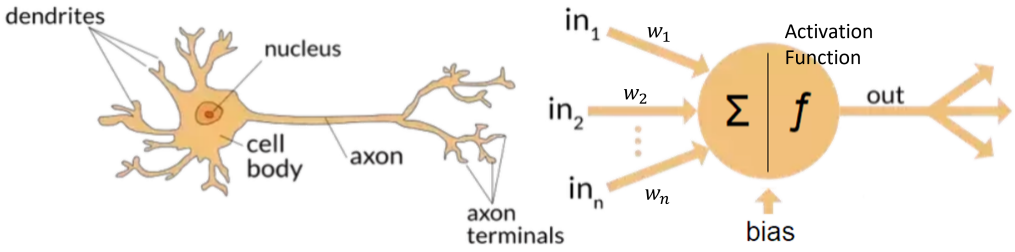

Figure 1.1: Resemblance between human neurons and network neurons

Connecting two neurons would form an **edge**. A parameter **weight** would often be attached to an edge and updated through the training process. Aggregated neurons would form a **layer**, different layer may perform different transformations on the input.

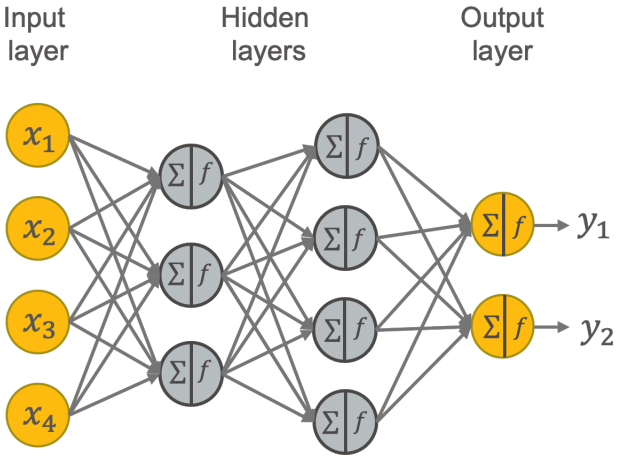

Figure 1.2: A simple example of neural network

1.3 Aims and Objectives

The main aim of the project is to build a regression model that uses ligand data as input to predict the binding affinity between different molecules. The output of this model can in turn be used to predict the possible rate of chemical reaction and important parameters guiding later experiments or researches.

The key objectives in this project are:

1. Investigate current state of art chemical approaches for predicting binding affinity and machine learning approaches applied to the same problem and their result.
2. Examine the former related work for their pros and cons.
3. Collect and generate relevant data for model training.
4. Design and develop a regression model using CNN.
5. Evaluate and optimize the model.
6. Create visualizations of model to provide a new perspective in understanding of how different parameters affect the reaction.

1.4 Description of the work

Comparing to all related work listed above, this project uses a new set of input data (Molecular Interaction Field) as features to describe the binding process. A regression model is developed to predict the binding affinity values of different protein-ligand interactions. Different CNN architecture (VGG16, Inception, ResNet) are implemented and tested over the new input data. Experiments are designed to find the optimal settings for each model and the hyper-parameter choices. The optimal model achieved a MAE of 0.07 given the accurate magnitude of concentration needed for a binding process and with a R^2 of -0.1.

1.5 Structure of this dissertation

In the following dissertation, I would first give an overview of the related work on this problem in Chapter 2: Related Work, bringing context about the researches done in the area of binding affinity prediction and related convolutional neural network architecture. Then in Chapter 3: Methodology, I discuss how I dealt with the dataset, how CNN is chosen as the main model, how metrics are chosen and which software to use. In the following Design chapter, I introduce more details about the convolutional neural network's terminology, the model's architecture design and the experiments to be carried out in the following chapter Implementation. The Implementation chapter would provide more details on the training and experiment process and the reasoning behind it. In next chapter Evaluation, the results of the experiments and the trained models are listed and discussed. Finally in chapter Reflection, I discussed about the project's plan, the progress of it in different terms, the contributions I made in this project, possible future extensions on this project's basis and a general reflection of the entire project.

Chapter 2

Related Work

This section will cover the following topics:

1. Former practices using CNN to predict binding affinity. - Section 2.1
2. Different CNN architectures in image classification area. - Section 2.2
3. Training tuning techniques. - Section 2.3

2.1 Predicting Binding Affinity with CNN

There are many literature doing the similar research on the issue of predicting binding affinity. In this section, some representative literature will be listed and reviewed based on their general approach and performance.

Kundu et al. provided a solution using random forest method and Gaussian process regression algorithm for prediction. They used the data from PdbBind Database and extracted structural information as features to train the model. With decent tuning and optimizing, this achieved the best result among all the former researches (before 2018) using the metric of *correlation coefficient* (R^2) and *root mean square error* (RMSE), providing a mature model without using neural networks [8]. These metrics are used later to evaluate the model's performance in similar researches. The success of this research provided a justification for the relationship between structural information and binding affinity, which sets the basic background of this project.

Ragoza et al. suggested a simple CNN architecture composed of only 3 convolutional layers with pooling layers for finding scoring functions used in drug discovery. Even with a simple architecture design and a shallow network, the model still performed well. This inspires the development of my baseline model with similar complexity. Though the trained model's output is not so relevant to this project, the comprehensive result of optimizations on the proposed CNN is helpful in guiding the implementation process [5].

Jones et al. provided a comprehensive study over the 3D-CNN and spatial graph neural network (SG-CNN) on this prediction problem applied in former literature. A fusion model together with 3D-CNN and SG-CNN was proposed with higher accuracy and greater computational efficiency [9]. The SG-CNN is not considered in this project as the feature input of this model is supposed to be extracted by CNNs.

The most relevant one is from Kwon et al., which provided an best performing model using 3D convolutional neural network[10], which is inspired by a novel CNN architecture - ResNet [11]. This architecture design introduces a shortcut connection, which imports the knowledge from former layers, significantly reduces the number of parameters to achieve similar results compared to other architecture. An ensemble-based approach with multiple predictors were applied to different network's result and enhanced the prediction quality. Their model was evaluated using Comparative Assessment of Scoring Functions 2016 (CASF-2016) dataset and demonstrated outstanding performance.

2.2 CNN architecture

Besides literature in computational chemistry field, implementations of different CNN architecture in computer vision field have been researched to guide the development of the model. Based on the performance on image classification tasks and the migration ability of these architectures, different models are implemented and experimented.

In 1998, Lecun et al. first brought up the concept of convolutional neural networks [12]. The LeNet implemented in this paper demonstrates the CNN's potential in replacing manually designed feature extractors, which initiates an exploration of CNN in the following decades.

Later in 2012, Krizhevsky et al. achieved top-1 error rate using CNNs in the ImageNet LSVRC-2010 contest, which is significantly better than the previous state of arts [13]. Besides its outstanding performance, the accurate feature extracted by the hidden layer is what proves its migration potential to other problems. They also imported ReLU activation function which accelerates the training process.

In another ImageNet contest in 2014, Simonyan and Zisserman examined closely of deep convolutional neural networks with a small convolution filters [14] and optimized previous architecture by increasing the depth of network. Their study of depth on CNN architecture, together with Zeiler and Fergus's study [15] on visualization of CNN's hidden layers, provides a deeper understanding of convolutional neural networks and each layer's functionality.

After familiarised with basic CNN architecture presented by Krizhevsky et al., researchers began modifying the building block of CNN, like Szegedy et al. did in Inception Net [16] and He et al. [11] did in ResNet. In Inception Net, a new block combining different sizes of convolutional filters have been aggregated to strengthen the ability of feature extraction. In ResNet, a shortcut connection is introduced to provide cross-layer information.

2.3 Model training and tuning

Model's training and tuning has been a well studied problem. Two factors found to be relevant to this are optimizers used and the architecture.

With some background research, first-order optimization algorithm proves to be the efficient and practical choice. Rudenko et al. provided a comprehensive overview of the first-order optimization algorithms [17]. And the Adam algorithm propose by [18] Kingma and Ba was studied.

This article provides a technique for accelerating the training by introducing batch normalization layer [19] and the technique is later applied to the ResNet architecture optimization.

Chapter 3

Methodology

3.1 Data Preparation

3.1.1 Feature data

Feature data is used for extracting useful information when training neural networks. In this project, feature data refers to the molecular interaction fields (MIF) values. Traditionally, CNNs take images or videos as input, where the input contains matrix of red, green, and blue values (RGB). To handle this problem, I discretize the MIF data into a grid. The generation process of data includes 4 main stages.

Firstly, the original data was generated as part of the publication [20] and re-used in this project. The reused data consists of 72 trajectories, demonstrating interaction of different ligands and a selected protein, which is the BD1 binding pocket of the bromodomain-containing protein 4 (BRD4).

Secondly, the generation of the trajectory data involves docking in the Molecular Dynamics field, which is beyond the scope of this project, more details could be found in relevant literature [21] . In summary, the docking searches the different conformations of a molecule that performs best in interaction with the target compound. Different conformations could be achieved by rotation, where Figure 3.1 shows all the conformers for one protein-ligand combination. The generated data includes 1000 snapshots of various conformers of a ligand.

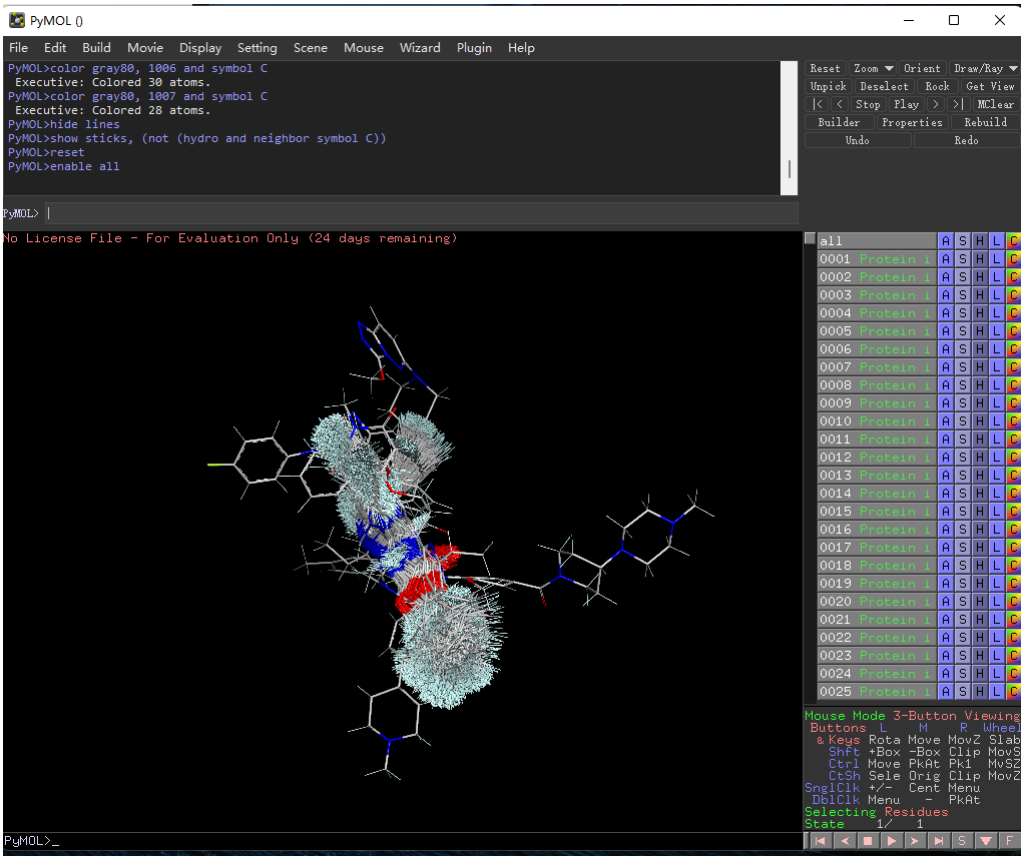

Figure 3.1: Sample protein-ligand conformers

Thirdly, with the aligned trajectory data, Molecular Interaction Field (MIF) data is then calculated using software opend3DQSAR [22] on each snapshot. In particular, steric MIF and electrostatic MIF were calculated following the Merck Molecular Force Field 94 (MMFF94) algorithm [23]. The first filed is electrostatic field, which is computed using Coulomb's law,

$$E_{ele} = k \sum_{i=1}^{n} \left[\frac{q_i}{r_i^2} \right]$$

where k represents the Coulomb constant, q_i stands for the charges of different objects and r_i is the distance between charges.

The second field describes the steric interactions by the van der Waals (vdW) energy,

$$E_{vdW} = \sum_{i=1}^{n} \left[\epsilon_{ij} \left((\frac{1.07 R_{ij}}{r_{ij} + 0.07 R_{ij}})^7 (\frac{1.12 R_{ij}^7}{r_{ij}^7 + 0.12 R_{ij}} - 2) \right) \right]$$

where r_{ij} stands for the inter-atomic distance, R_{ij} for the minimum energy separation in $\mathring{A}$ and ϵ_{ij} is the well depth parameters.

After examining all the trajectories, the minimum box used to wrap all the conformers got the shape $(109, 104, 85)$. The box's grid step size was set to $0.333\mathring{A}$ to maintain precision. The generation script used could be found in this script.

After generation of field data, scripts have been written to extract the exact field values and transform the data into *.npy* format to increase the loading speed and save disk space.

3.1.2 Label data

Label data is used for validating model's output and adjusting parameters. In this project, label data is the binding affinity value for each protein-ligand trajectory. As discussed in Section 1.2 - Background, binding affinity provides a measurement of the concentration needed to inhibit a biological process. The data was collected from ChEMBL database (95%) [24] and PdbBind-CN database [25, 26] and persevered in terms of pIC_{50}.

3.2 Model choice

After pre-processing and gaining more knowledge about the training data, the problem could be defined more clearly as a regression problem with 3D grid data input. The massive amount of high dimensional data available and the lack of explicit features brought my attention to deep learning area, where features could be extracted automatically by the multi-layer neural network.

As described in Section 2.1, CNN has been applied widely on this problem and showed promising result. Now with a similar objective, it is worthwhile to try applying CNN on this new dataset.

Besides experience in computational chemistry field, CNN has been studied and attempted in computer vision field with great many of success cases, especially in image recognition tasks [5]. Former research have been studied (in Section 2.2) and the CNN's ability to generalize well on other datasets, as indicated by Zeiler and Fergus [15], motivates me to apply CNN on this problem.

3.3 Metric choice

Metrics are used to evaluate the model's prediction against the actual values. In Tensorflow libraries, these are referred to as loss functions during training. With study of related work on this problem and the nature of this regression problem, metrics considered includes:

1. Mean Square Error (MSE) - Computes the mean squared error between labels and predictions.
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{pred(i)} - y_{actu(i)})^2$$

2. Root Mean Square Error (RMSE) - Computes the root of mean squared error between labels and predictions.
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{pred(i)} - y_{actu(i)})^2}$$

3. Mean Absolute Error (MAE) - Computes the mean absolute error between labels and predictions.
$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_{pred(i)} - y_{actu(i)}|$$

4. Pearson correlation coefficient r- Computes the covariance between y_{pred} and y_{actu}.
$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

5. Coefficient of Determination R^2 - Gives a measurement about the goodness of fit of a regression model.
$$R^2 = 1 - \frac{\sum(y_{actu} - y_{pred})^2}{\sum(y_{actu} - \bar{y})^2}, \quad \bar{y} = mean(y_{actu})$$

MSE is used when training the model and its derivative RMSE is used when evaluating the model's performance on test sets. As the label data pIC_{50} represents the log value of concentration, MAE is also used to evaluate the model's performance on test set

as it indicates the concentration level needed. These two metrics gives an intuition of the absolute value the model predicts. Pearson correlation coefficient and coefficient of determination are used to evaluate how close the prediction value is to the actual value, providing another perspective in relative value when evaluating the model.

3.4 Software choice

The main software used during this project is Open3DQSAR [22] and Tensorflow [27] library.

Open3DQSAR is used to generate the Molecular Interaction Field (MIF) data using different trajectories and settings. It is open-source software and widely accepted in computational chemistry.

Tensorflow, as a mature machine learning framework, provides basic implementation of training components. By using it for model construction, I could focus more on the adjustment of network architecture, tuning of hyper-parameters and optimisation of training settings on a high-level perspective.

Chapter 4

Design

As this is more a research focused project, there is less emphasis on producing a design for complicate system. However, design choices have been made early based on literature review. In this chapter, CNN terminology would be explained, together with the design of CNN architectures and experiments.

4.1 CNN terminology

In this section, an overview of CNN specific terminology would be discussed. This includes the basic units of a CNN, like different layers, possible loss functions and optimizers. Some settings of these components may be modified in later optimization process, therefore it is necessary to give an brief explanation here.

4.1.1 Layers

Layers are the basic component of neural networks. In CNNs, some new types of layers were introduced. In this section, I would give an overview of each kind of layer on its functionalities and the possible optimization on each layer.

Convolutional Layer Convolutional layer servers as the feature extractor in CNN. In each layer, kernels are used to scan through the entire input tensor to get the eigenvalues of the small areas. For a kernel, its size, stride and initialization could be customized. The kernel size specifies the depth, height and width of the 3D convolution window. The stride specifies the units the kernel move along each spatial dimension. The initialization method would create differently weighted kernels at the beginning of training.

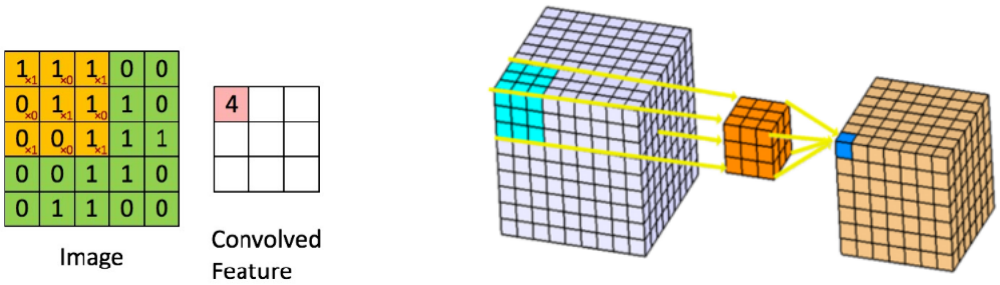

Figure 4.1: Left figure shows the first step of a convolution process on a 2D input. The yellow area resembles the kernel applied, and the value in the bottom right corner stands for the kernel weights. The red value stands for the current convolved value of kernel. Right figure shows the first step of a convolution process on a 3D input.

A filter is the collection of all kernels which convolved on the channels of the input tensor, which stacks the convolved features on the last axis. In other way, the number of filters applied determines the last dimension of the output tensor.

Pooling layers Pooling layer serves as the down-sampling unit in CNN, since the output of convolutional layers is still a large tensor. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. The adjustable settings are the pooling size and type. Two common types of pooling are in popular use: max and average. Max pooling uses the maximum value in the neuron cluster while the average pooling uses the average value.

Fully Connected Layer It is believed fully connected layers servers as the predictor in CNN. This kind of layers connect every neuron in one layer to every neuron in another layer. Each neuron has an activation function, which defines the signals this neuron would output with different input. Commonly used activation function includes logistic $\left(\sigma(x) = \frac{1}{1+e^{-x}}\right)$, hyperbolic tangent $\left(\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\right)$ and Rectified linear unit $(ReLU(x) = \max\{0, x\})$

Drop out layer The Dropout layer randomly sets input units to 0 with a frequency of rate (predefined hyperparameter) at each step during training time, which helps prevent overfitting.

Batch Normalization layer This layer performs a transformation that maintains the mean output to 0 and the output standard deviation to 1. According to Ioffe and Szegedy [19], using this technique could accelerate the training of deep neural network.

Zero Padding layer This layer would add additional values at the fringe of input grids to help the convolutional layer capture the potential features at the corner of the input grid.

4.1.2 Loss Function

Loss functions, as discussed in Metric choice, are the functions used to evaluate how well the model fits the data. The better the prediction the model made, the lower the loss function would output. In this project, the RMSE are chosen as the main loss function during training following the common practice in regression problem.

4.1.3 Optimizers

Optimizers are algorithms or methods used to minimize the loss function. They are mathematical functions independent on model's learnable parameters (like weights or bias on neurons). There are many mature optimizers and have different advantages and disadvantages. Broadly speaking, optimization algorithms could be divided into two categories, first order and second order. As second-order algorithm is computationally expensive, most algorithms used in optimizing neural networks is the first order algorithm based on gradient descent. In general, a first-order optimization algorithm would first calculate the gradient of current parameter, based on the history gradient calculate the first-order and second order momentum, then calculate the gradient descent for this moment and finally update the parameter.

Classical optimizers includes:

- **Gradient Descent (GD)** - Basic optimization algorithm, which reduces the loss function by moving in the direction opposite to the steepest ascent. It is easy to implement but computationally expensive as the gradient was calculated from the entire dataset.

- **Stochastic Gradient Descent (SGD)** - A variant of GD algorithm, which calculates the gradient based on a randomly selected subset of the data. It requires less memory, converges in less time but may result in high variance of the model.

- **SGD with momentum (SGDM)** - A variant of SGD algorithm, which includes the momentum mechanism that remembers the update value δw at each iteration and determines the next update based on a linear combination of the gradient and previous update. The introduce of momentum helps to achieve faster convergence with less oscillations and solve the high variance problem. However, it introduces another hyper-parameter (momentum) to be tuned.

- **Adagrad** - Stands for Adaptive gradient descent algorithm. Unlike the algorithms above, Adagrad first introduces the concepts of updating the learn-

ing rate for different parameters using second-order momentum. This strategy would increase the learning rate for sparser parameters and decrease the learning rate for less sparse ones. It is proved to be more effective where data is sparse. For some cases however, as the second-order momentum is monotonically increasing, the learning rate for certain parameters may be decreased to 0 quickly, this stops the update and terminates the training process.

- **Root Mean Square Propagation (RMSProp)** - This optimization algorithm modifies the way Adagrad used to update the learning rate. In stead of focusing on the cumulative sum of squared gradients, RMSProp uses a running average of the magnitudes of recent gradient to update the learning rate.

- **Adaptive Moment Estimation (Adam)** - In this optimization algorithm [18] the advantages of former algorithms are combined. It uses the first-order momentum from SGDM and the second-order momentum from RMSProp. It converges rapidly, rectifies vanishing learning rate problem and solved the high variance of trained model.

4.2 Model Design - Architecture

As mentioned in CNN architecture, different architecture was implemented and adapted for this problem. These architectures includes a baseline model, a VGG net model, an Inception Net model and a ResNet model. The detailed design of model code could be found in the related git repository.

The **baseline model** stems from AlexNet [13], it uses the combination of convolutional layer and pooling layer to extract the features and two fully connected layers are used to make the final prediction. This model is used to set the minimum expectations on the performance. Tuning of parameters and training settings are first applied on this model to estimate the possible improvements.

The **VGG model** took inspiration from VGG net [14], which increases the depth of the convolutional network and improved the performance. This model is an extension of

the baseline model with increased depth.

The **Inception net model** originates from [16], which introduces a new structure to help with the feature extracting process. This model adapts the idea to this problem and uses an ensemble of convolutional layers in the feature extraction process.

The **ResNet model** stems from ResNet [11], it uses a new residual block to bring prior knowledge from former layers to ensure the increase of network depth could improve the model's result.

4.3 Experiments

Experiments are designed to compare the influence of different training settings, the performance of different architecture and find the optimal settings for different models.

The training settings to be experimented includes:

- train set selection methods
- train-test split methods
- optimizer choice and tuning
- batch size choice

The different architecture refers to subtle changes to the architecture proposed in previous section. This subtle changes includes:

- The number of filters in convolutional layers
- The type of pooling layers
- The number of fully connected layers
- The activation function used in fully connected layers

The different models to test are mentioned in the previous section.

Details of experiments could be found in next chapter.

Chapter 5

Implementation

In this chapter, the implementation details of different CNN architecture and the reasoning behind the experiments design would be discussed. Due to the late acquisition of the training set, not all experiments designed are performed.

5.1 Architecture Details

Baseline - This model contains two 3D convolutional layers. Each convolutional layer is followed by a max pooling layer. The parameters in the last pooling layer are then flattened and feed into a neural network with two fully connected layers.

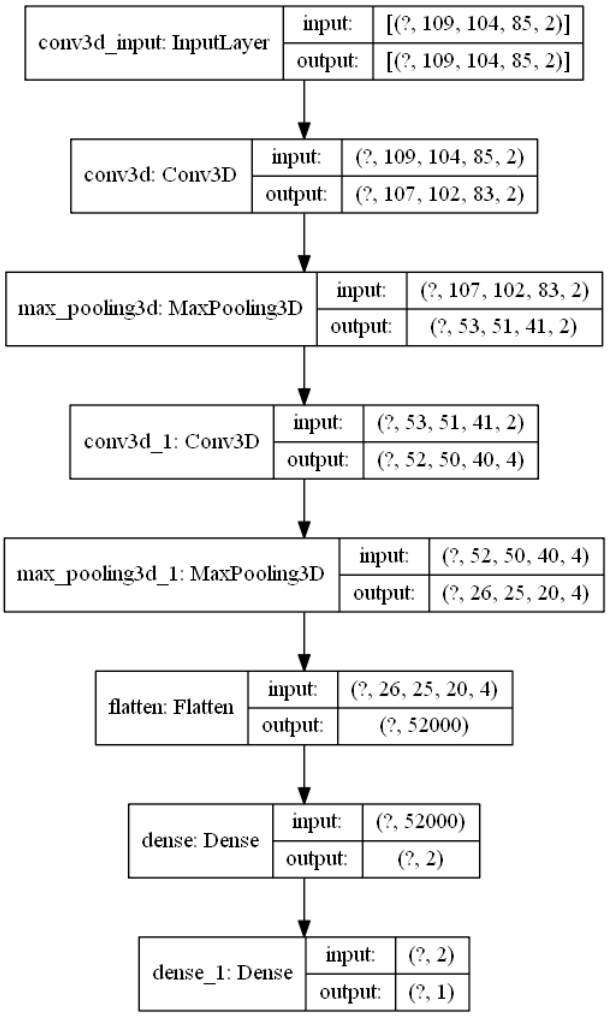

Figure 5.1: baseline architecture

VGG-net - This model mimics the architecture design of VGG16 model (configuration C) in [14].

Inception Net - This model mimics the design of Inception model in [16].

ResNet - This model mimics the architecture design of the ResNet34 model in [11].

Figures of last three architectures' detailed layer configuration could be found in Appendix.

5.2 Training settings

Further to Feature data, two separated field data from the same snapshot are combined into one frame. To help reproduce the experiments, the training dataset is shuffled with the tensorflow random seed set to 12. The train-test split on different trajectories is also fixed and saved to reduce the time used to process and read the data.

5.2.1 Training set selection

As the whole training set is too big, it is computationally impossible to perform the training on the entire training set, therefore a subset need to be selected. Two selection methods are proposed, the first is read in a set of continuous frames and the second is read in a set of discontinuous frames with a pre-defined step size. For example, the first method would read the frames with index 1 to 50, while the second method would read the frames with index 1,21,41,... using a step size of 20.

5.2.2 Train test split

Two forms of train-test split is proposed. The first is molecule-wise and the second is frame-wise. Suppose the dataset contains 10 protein-ligand binding interactions, each with 1000 frames, the first method would divide 8 interactions into the training set and the rest interactions to test set, while the second method would extract 800 frames from each interaction into the training set and the rest frames to test set.

According to the model's design purpose - predicting the binding affinity on unseen protein-ligand binding interaction, molecule-wise method is chosen to perform the train-test split. Following the common practice, the proportion of test set are set to 20% of the entire dataset.

5.2.3 Optimizer Choice

According to former research, the choice of optimizer has an impact on the training speed and how parameters are updated.

Due to the limited time for experiments, Adam is chosen as the main optimizer as it is both computationally efficient and converges rapidly.

5.2.4 Batch size

In modern machine learning, batch size is used to define the number of samples to work through before updating the internal model's parameter. Different batch size would effect the model's training time and it's performance. Larger batch size is more likely to find the global optima with a higher memory cost while the smaller batch size tend to have a faster converge speed [28].

5.3 Change to architecture

Following the background research, changes to the CNN architecture could influence the training outcome. Experiments have been carried out to test different settings' influence.

Convolutional layer - In each convolutional layers, the number of filters and the size of kernels could be changed. According to experiments, larger number of filters tend to perform well as it probably preserve more features. The kernel size is best set to 2 or 3 to capture the features.

Pooling layers - Different polling type could be applied to extract the features from the input data. In this project, the pooling layers used are mainly max pooling.

Chapter 6

Evaluation

6.1 Results and interpretation of Experiments

For the experiments on the training set selection methods, little difference is observed in terms of the training speed and the model's performance.

For the experiments on the training batch size selection, an interesting discovery was made. As in Figure 6.1, larger batch size tend to converge slower than smaller batch size. The batch-32 experiment converges at around 5 epochs while more than 10 epochs are used for other two groups. In the meanwhile, large batch size's model performance tend to oscillate a lot while small batch size seems more stable. According to the experiments, small batch size seems to be a better choice when training the model.

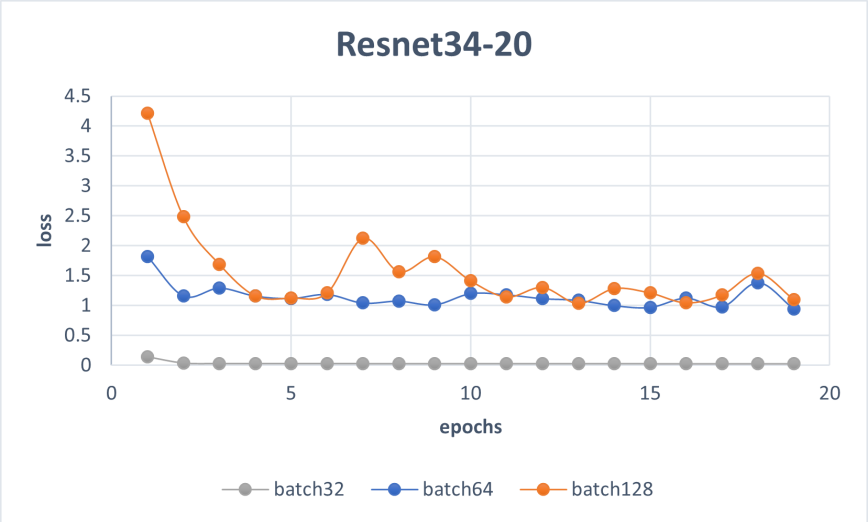

Figure 6.1: Different batch size on ResNet34 model.

For different models, due to the limited training time, could only be trained with 20 or 50 instances training set each. The training time each epoch costs and the optimal results during the training are listed in the table.

	Avg.	Train			Test		
Model	Epoch time	MAE	MSE	R^2	MAE	MSE	R^2
Baseline-20	3min	2.272	7.037	-6.016	1.714	4.048	-17.103
VGG16-20	98min	0.781	0.998	-0.034	0.521	0.378	-0.690
Inception-50	16min	0.167	0.045	-0.453	0.073	0.008	**-0.101**
ResNet34-50	70min	0.172	0.048	N/A	0.089	0.012	-0.799

Table 6.1: Results of optimal models trained. The model is denoted as (architecture name)-(training samples count). The optimal model trained - the Inception-50 model is marked bold in terms of R^2.

From the above table, it could be easily found that all models with deeper network or more complex architecture outperforms the baseline model, which shows deeper network did extracted some useful features. Generally speaking, all models other than baseline could accurately predict the magnitude of the binding affinity, which gives an MAE below 1. This could be inspirational for estimating a new protein-ligand's binding interaction. Comparing different model's training time, the Inception model seems to be the most time-efficient while achieving the best results. I suppose the inception block using multiple convolutional layers successfully extracted more valuable features than other architectures.

Chapter 7

Reflection

7.1 Project management

7.1.1 General Plan

The entire project could be divided roughly into three stages.

Stage 1 is doing literature research to gain background knowledge in the relevant chemistry field and explore former related work. This stage involves communication with the chemistry professionals to understand the background, obtaining the data, researching former work.

At the end of stage 2, processing data, developing and optimizing the model should be completed. This stage involves reproducing former studies on this field. A CNN model with novelty in training process and architecture should be developed. Along with the development, a formal essay (If the model runs well) describing CNN's designed architecture, the training algorithm, the findings with the trained model and possible improvements of this model should be finished.

Stage 3 is to prepare for the final dissertation deliverable. During this stage, the thesis should be finished and demo should be prepared. If there are time left, more research on some trivial branches of the former model could be done in order to find possible improvements.

The plan below is a revised version compared to the one in the proposal. Due to the fact that relevant research data provided by other research fellow was obtained late, the stage 2 of development could only be postponed.

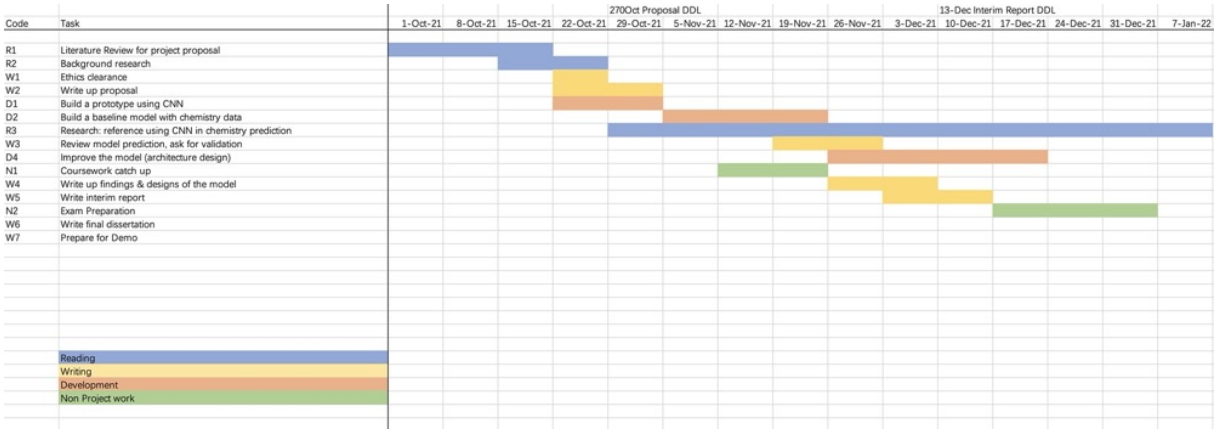

Figure 7.1: Proposal plan

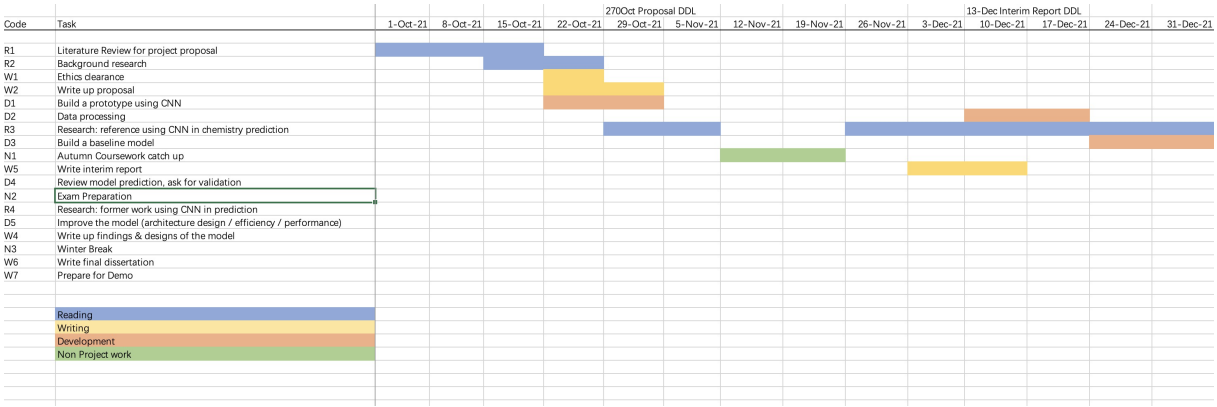

Figure 7.2: Current plan - Part 1

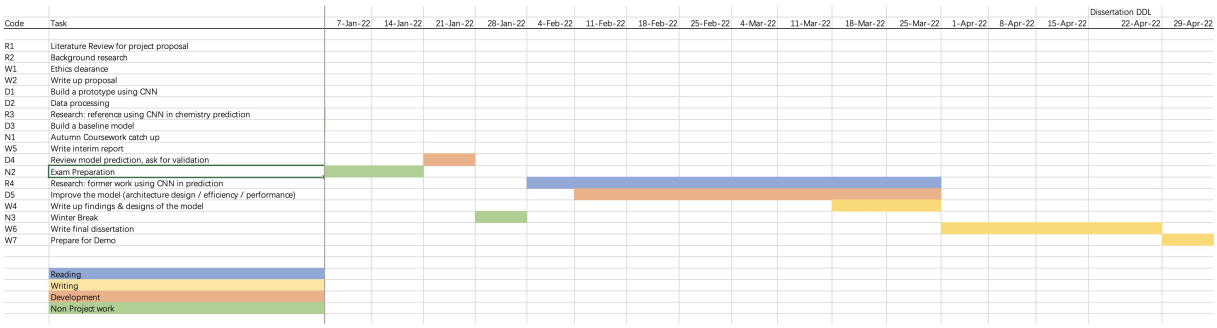

Figure 7.3: Current plan - Part 2

7.1.2 Progress

In the first term, I have been able to carry out research in chemistry field with machine learning techniques applied. An overview of machine learning's application has been made, the significance of statistical approach in chemistry has been recognised. The reviewed literature could help with the later implementation stage. Whilst a lot of literature review work have been done, overall the project is not where I had hoped it would be. The delay of the chemistry data caused a lot development work to be postponed. Meanwhile, doing a 70/50 credit split with a 100% coursework assessed module significantly affected my progress. It is not surprising that I am still at an early stage of implementation.

In the second term, I focused more on the implementation work, completing the data pre-processing, coding and adapting different models and carrying out the experiments. I implemented the data pre-processing pipeline as soon as I got the original data. The massive amount of data (1.2 TB in total) took a long time to generate and transformed into a format friendly to disk storage and training. Though I implemented different architectures during the processing of the data, it still leaves small amount of time (approximately three weeks) to perform experiments on different models with different settings. Only a small proportions of designed experiments could be carried out.

7.2 Contributions

In summary, as a research project, this project explore the potentials of using different CNNs to predict the binding affinity with molecular interaction field data and completed some foundation work for future endeavors. The project shows promising result as the Inception model performs an approximate prediction with a R^2 around -0.1 even with a limited training instance used. The project also adapted different architecture design from image classification tasks to this regression problem. Massive molecular interaction field data (1TB) are generated and processed ready to be used for later training. With experiments, this project also shows that tuning of the model could have a strong impact and good settings or hyper-parameter choices are found to either accelerate the training

or optimize the training result.

7.3 Future work

There is large scope for future work on many aspects of the project.

The obvious next steps include continuous training and hyper parameter tuning. It is probable that better performance could be gained with a tuned model. For example, a combination of normal convolution and dilated convolution could be used in Inception model during the feature extraction process. Additionally, it could also be worthwhile training the model on larger scale (only 50 frames per interaction are used in this project's models).

Longer term expansion of the project includes implementing more novel techniques in machine learning like the transformers and the attention block mechanism. As the researchers explore in this problem more, it is likely mature solutions would be proposed for this problem. Another possible expansion is doing more experiments to find the best architecture design. In image classification field, an architecture would have different variants (like ResNet has ResNet34, ResNet50, etc.) and with fine tuning the deeper the network the better performance could be achieved. Most models proposed and trained in this project is still at an early stage of this kind of tuning and has great potential in improving the performance.

Other more theoretical areas include exploration of the model's internal representations used to predict the binding affinity. As [13] did in the AlexNet model, the feature vectors from the last hidden layer was extracted and visualized, which demonstrate the capabilities of CNN in extracting features from images. Similarly, such visualization could be implemented for the molecular interaction field, exploring the key spatial areas effecting the prediction process and could possibly guide the research in deeper understanding of MIF.

Another area of exploration is reviewing how well the model performs on unseen-datasets. Due to the limitation of knowledge background as a computer science student, it is hard to find and use relevant datasets for testing with a similar input and output of

this project. If the model performs well on a broader term of predicting protein-ligand binding process, it could be adapted to help with the drug discovery.

7.4 Self Reflection

The scope of project I proposed was ambitious, and the goals was not clear until the late first term. This was due to the lack of knowledge with the data and the late acquisition of training data. As the majority of the implementation work began after the January exams, it quickly became apparent that I could not run all the experiments within the given time so the scope of the project was narrowed. This was mainly due to my underestimation of the time spent on data pre-processing.

In spite of the unexpected long time used for data preparation, my personal time management was for the most part successful. All key dates and deadlines were met throughout the year in addition to managing coursework for other modules and applying for master programs. However, occasionally I tend to spend time exploring tangential areas following the lead of project, like different optimizers' evaluation. Though this exploratory work may prove useful in the long run, it had very little effect on the completion of the project.

Overall the project was a stressful but rewarding experience. I have been able to learn many new techniques and concepts in chemistry and machine learning, particularly deep learning field, as well as developing my coding skills implementing all the different models and experiments. I am happy where the project takes me to.

Bibliography

[1] Johan Perols. "Financial Statement Fraud Detection: An Analysis of Statistical and Machine Learning Algorithms". In: *AUDITING: A Journal of Practice & Theory* 30.2 (May 1, 2011), pp. 19–50. ISSN: 0278-0380. DOI: 10.2308/ajpt-50009.

[2] Robert Culkin and Sanjiv R Das. "Machine Learning in Finance: The Case of Deep Learning for Option Pricing". In: (), p. 15.

[3] Nongnuch Artrith et al. "Best Practices in Machine Learning for Chemistry". In: *Nature Chemistry* 13.6 (June 2021), pp. 505–508. ISSN: 1755-4349. DOI: 10.1038/s41557-021-00716-z.

[4] Alexandre Tkatchenko. "Machine Learning for Chemical Discovery". In: *Nature Communications* 11.1 (Aug. 2020), p. 4125. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17844-8.

[5] Matthew Ragoza et al. "Protein–Ligand Scoring with Convolutional Neural Networks". In: *Journal of Chemical Information and Modeling* 57.4 (Apr. 2017), pp. 942–957. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.6b00740.

[6] Volker L. Deringer et al. "Realistic Atomistic Structure of Amorphous Silicon from Machine-Learning-Driven Molecular Dynamics". In: *The Journal of Physical Chemistry Letters* 9.11 (June 2018), pp. 2879–2885. ISSN: 1948-7185, 1948-7185. DOI: 10.1021/acs.jpclett.8b00902.

[7] Stefano Curtarolo et al. "The High-Throughput Highway to Computational Materials Design". In: *Nature Materials* 12.3 (Mar. 2013), pp. 191–201. ISSN: 1476-1122, 1476-4660. DOI: 10.1038/nmat3568.

[8] Indra Kundu, Goutam Paul, and Raja Banerjee. "A Machine Learning Approach towards the Prediction of Protein–Ligand Binding Affinity Based on Fundamental Molecular Properties". In: *RSC Advances* 8.22 (2018), pp. 12127–12137. ISSN: 2046-2069. DOI: 10.1039/C8RA00003D.

[9] Derek Jones et al. "Improved Protein–Ligand Binding Affinity Prediction with Structure-Based Deep Fusion Inference". In: *Journal of Chemical Information and Modeling* 61.4 (Apr. 2021), pp. 1583–1592. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.0c01306.

[10] Yongbeom Kwon et al. "AK-Score: Accurate Protein-Ligand Binding Affinity Prediction Using an Ensemble of 3D-Convolutional Neural Networks". In: *International Journal of Molecular Sciences* 21.22 (Nov. 2020), p. 8424. ISSN: 1422-0067. DOI: 10.3390/ijms21228424.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385 [cs].

[12] Y. Lecun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.

[14] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556 [cs].

[15] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *arXiv:1311.2901 [cs]* (Nov. 2013). arXiv: 1311.2901 [cs].

[16] Christian Szegedy et al. "Going Deeper With Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[17] Oleg Rudenko, Oleksandr Bezsonov, and Kyrylo Oliinyk. "First-Order Optimization (Training) Algorithms in Deep Learning". In: (), p. 15.

[18] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980 [cs].

[19] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, June 2015, pp. 448–456.

[20] Ellen E. Guest, Stephen D. Pickett, and Jonathan D. Hirst. "Structural Variation of Protein–Ligand Complexes of the First Bromodomain of BRD4". In: *Organic & Biomolecular Chemistry* 19.25 (June 2021), pp. 5632–5641. ISSN: 1477-0539. DOI: 10.1039/D1OB00658D.

[21] Veronica Salmaso and Stefano Moro. "Bridging Molecular Docking to Molecular Dynamics in Exploring Ligand-Protein Recognition Process: An Overview". In: *Frontiers in Pharmacology* 9 (2018). ISSN: 1663-9812.

[22] Paolo Tosco and Thomas Balle. "Open3DQSAR: A New Open-Source Software Aimed at High-Throughput Chemometric Analysis of Molecular Interaction Fields". In: *Journal of Molecular Modeling* 17.1 (Jan. 2011), pp. 201–208. ISSN: 0948-5023. DOI: 10.1007/s00894-010-0684-x.

[23] Thomas A. Halgren. "Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94". In: *Journal of Computational Chemistry* 17.5-6 (1996), pp. 490–519. ISSN: 1096-987X. DOI: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[24] Anna Gaulton et al. "The ChEMBL Database in 2017". In: *Nucleic Acids Research* 45.D1 (Jan. 2017), pp. D945–D954. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkw1074.

[25] Renxiao Wang et al. "The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures". In: *Journal*

of Medicinal Chemistry 47.12 (June 2004), pp. 2977–2980. ISSN: 0022-2623. DOI: 10.1021/jm0305801.

[26] Renxiao Wang et al. "The PDBbind Database: Methodologies and Updates". In: *Journal of Medicinal Chemistry* 48.12 (June 2005), pp. 4111–4119. ISSN: 0022-2623. DOI: 10.1021/jm048957q.

[27] *TensorFlow*. https://www.tensorflow.org/.

[28] Leslie N. Smith. "A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay". In: *arXiv:1803.09820 [cs, stat]* (Apr. 2018). arXiv: 1803.09820 [cs, stat].

Appendix A

Additional Figures

In all figures below, the convolutional layers are denoted as (kernel size,conv, filters, strides), max pooling layers are denoted either through line (pool, /pool size) or block (max pool, /pool size), fully connected layers are denoted as (fc, neuron count), drop out layers denoted as (drop out, rate).

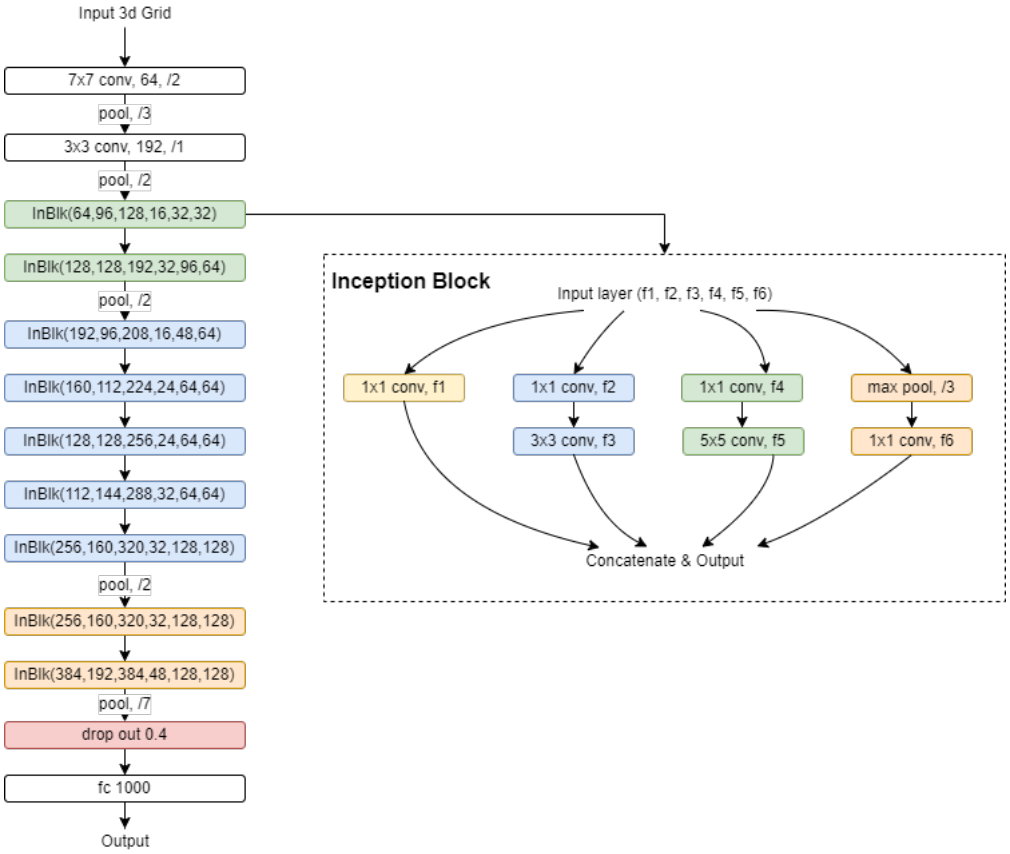

Figure A.1: Architecture of Inception

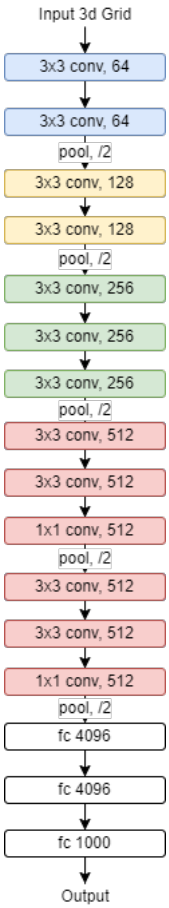

Figure A.2: Archiecture of VGG16

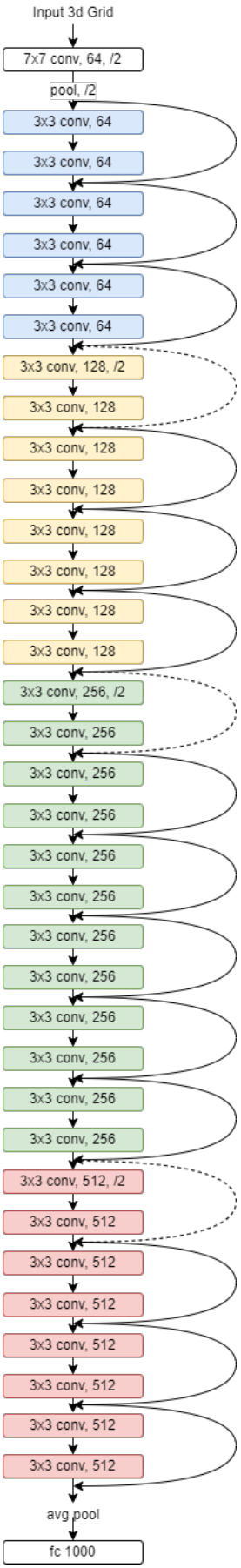

Figure A.3: Architecture of Resnet34