Predictive Modeling of Small Molecule Binding Affinity through a NequIP-based Equivariant Neural Network

Vladislav Cherdantsev

1 Introduction and Background

Accurately predicting protein-ligand interactions is critical in drug design where even small improvements can lead to significant advancements in therapeutic development. In recent years, the integration of deep learning models has emerged as a promising approach to address this challenge.[1] Leveraging 3D structures of protein-ligand interactions as input data, these models have shown potential in providing precise predictions of binding affinity, offering a cost-effective and efficient alternative to traditional experimental assays.[2]

Despite significant advancements, certain areas of drug discovery still face great challenges. Tuberculosis (TB) drug discovery, for instance, has encountered limited success over the past five decades, yielding only two FDA-approved drugs. The development of effective treatments for TB is particularly challenging due to factors such as drug resistance and the complex biology of the *Mycobacterium tuberculosis* bacterium. One potential avenue for TB drug discovery involves targeting specific proteins within the bacterium. For example, β-ketoacyl synthase KasA has emerged as a promising target with a characterized bound inhibitor structure.

In an effort to improve the predictive accuracy of small molecule binding affinity and address the challenges in tuberculosis (TB) drug discovery, this study introduces an equivariant neural network based on NequIP[3] that predicts pharmacokinetic data pertaining to a series of congeneric ligands targeting KasA. Training of this model is conducted on a dataset comprising approximately 300 small molecules with known binding poses. These poses were obtained through biological assays, mapping to known binder structures, and molecular dynamics (MD) energy minimization. More specifically, the neural network predicts the area under the concentration-time curve (AUC), a pharmacokinetic metric that quantifies the total exposure to a drug over a specified time period, from the 3D structures of the ligands.

2 Related Work and Methods

NequIP

For this project, I selected NequIP[3] as the base model. The architecture of NequIP is illustrated in Figure 1.

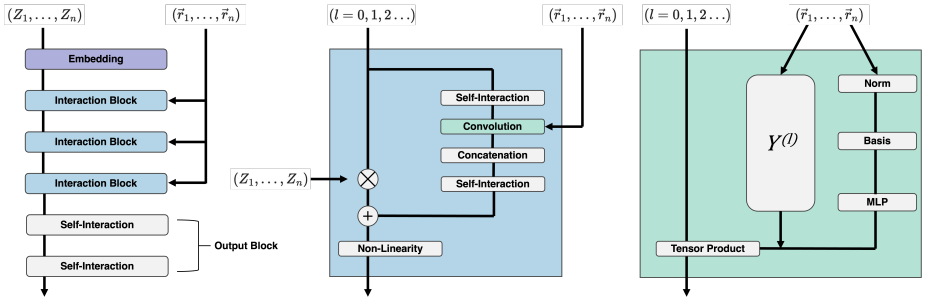

Figure 1: (A) The NequIP network architecture. (B) The structure of the Interaction Block. (C) The structure of the Convolution Layer.

The NequIP model takes as input 3D molecular graphs, which are characterized by a set of atomic numbers $\{Z_i\}$ and atomic positions $\{\vec{r}_i\}$. The model aims to predict a scalar value representing the total potential energy of the system, computed as the sum of atomic potential energies:

$$E_{pot} = \sum_{i \in N_{atoms}} E_{i,atomic}$$

Here, the per-atom energies $E_{i,atomic}$ are the scalar outputs predicted by an equivariant message-passing neural network, which will be further described. Despite the predicted potential energy being invariant under translations, reflections, and rotations, the network incorporates internal features that transform as geometric tensors and are equivariant to rotation and reflection, making the model more expressive.

Embedding

The forward pass of the data starts with the Embedding block (Figure 2). Within this block, the atomic numbers (a categorical feature) are one-hot encoded and then passed through an Atomwise Linear layer, which transforms the encodings into embeddings, usually of higher dimensions. These chemical embeddings derived from the atomic numbers serve as initial node features, with their dimensionality determined by the multiplicity of the node features' scalar irreps, which is a hyperparameter we can adjust. For example, setting the number of node feature irreps that transform as scalars to 32 (denoted as '32x0e' in the e3nn[4] notation) would yield output embeddings with a dimensionality of 32 for this Atomwise Linear layer.

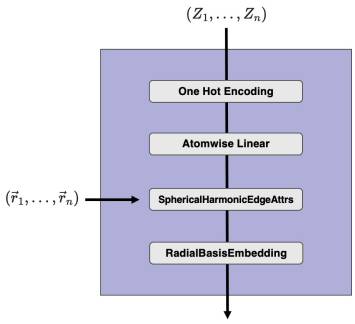

Figure 2: The structure of the Embedding Block

In the next step, the edge vectors $\{\vec{r}_{ij}\}$ are derived from the atomic positions $\{\vec{r}_i\}$ and then processed through a SphericalHarmonicEdgeAttrs layer to obtain the spherical harmonic projections $Y_m^{(l)}(\hat{r}_{ij})$ of the edge vectors up to the maximum rotation order l_{max}, which can be adjusted as a hyperparameter. Additionally, the edge lengths, calculated as the magnitudes of the edge vectors, are fed into a trainable RadialBasisEmbedding layer denoted as B to generate basis embeddings of the interatomic distances. The dimensionality of this basis embedding, denoted as N_b, is a hyperparameter typically set to 8. The basis embeddings in NequIP are derived from radial Bessel functions and a polynomial envelope function f_{env}:

$$B(r_{ij}) = \frac{2}{r_c} \frac{\sin\left(\frac{b\pi r_{ij}}{r_c}\right)}{r_{ij}} f_{env}(r_{ij}, r_c) \in R^{N_b}$$

where r_{ij} represents the interatomic distance, while r_c denotes a local cutoff radius that confines interactions to atoms closer than a specified cutoff distance. The polynomial envelope function f_{env} is defined as:

$$d = \frac{r_{ij}}{r_c}$$

$$f_{env}(d, p) = 1 - (p+1)(p+2)d^p + p(p+2)d^{p+1} - p(p+1)d^{p+2}$$

with $p = 6$ operating on the interatomic distances normalized by the cutoff radius $\frac{r_{ij}}{r_c}$. At network initialization, the Bessel roots are set as $b = [1, 2, \ldots, N_b]$, and we subsequently optimize $b\pi$ via backpropagation rather than keeping it constant.[3]

Interaction Block

The next block of the model is the Interaction Block (Figure 1B), which starts with a Self-Interaction layer implemented through e3nn's Linear module[4] de-

signed to be equivariant to $O(3)$. This Self-Interaction block learns and uses different weights for different rotation orders $l = 0, 1, 2, \ldots$ (but the same weights are used for every m for a given order) to preserve the symmetry of the feature irreps. For example:

$$e3nn.Linear(\text{"}4x0e+16x1o\text{"}, \text{"}8x0e+8x1o\text{"}) = concat\left(W_1 \cdot input[:4],\, W_2 \cdot input[4:]\right)$$

In the first Interaction Block, where only $l = 0$ features are present (the chemical embeddings described earlier; e.g., "32x0e"), this 'e3nn.Linear' layer operates just like a standard linear layer and does not have any distinct functionality. However, as higher-order tensor features are introduced in subsequent blocks, the use of separate weights for different l values becomes crucial for maintaining equivariance throughout the model.

Convolution

Following the Self-Interaction layer, the Interaction Block has a Convolution layer (Figure 1C). Convolution operations inherently possess translation and permutation invariance,[3] but to achieve rotation-equivariant convolution, it's necessary to use rotation-equivariant filters. To ensure this property, we constrain the filters to be products of learnable radial functions and spherical harmonics:[5]

$$F_m^{(l)}(\vec{r}_{ij}) = R(r_{ij}) Y_m^{(l)}(\hat{r}_{ij})$$

The radial function $R(r_{ij})$ is realized through a multi-layer perceptron and outputs the radial weights for all subsequent filter-feature tensor production in 'e3nn.o3.TensorProduct':

$$R(r_{ij}) = W_n \sigma\left(\ldots \sigma\left(W_2 \sigma\left(W_1 B(r_{ij})\right)\right)\right)$$

Both $Y_m^{(l)}(\hat{r}_{ij})$ and $B(r_{ij})$ were obtained in the previous Embedding block. Filters following this structure inherit the transformation property of spherical harmonics under rotations because $R(r_{ij})$ is rotationally invariant, and all learnable weights in the filter belong to $R(r_{ij})$.

Finally, using these filters, the convolution combines the product of the radial function $R(r_{ij})$ and the spherical harmonics $Y_m^{(l)}(\hat{r}_{ij})$ with neighboring features in an equivariant manner via a tensor product, thereby generating more scalar and higher-order tensor features:[4]

$$f_i' = \frac{1}{\sqrt{z}} \sum_{j \in \mathcal{N}(i)} f_j \otimes \left(R(r_{ij})\right) Y_m^{(l)}(\hat{r}_{ij})$$

where:

- f_i' are the updated node features
- f_j are the input features of the neighboring nodes
- $\mathcal{N}(i)$ is the set of neighbors of the node i
- z is the average number of neighbors of a node
- $R(r_{ij})$ are the radial weights described above
- $a \otimes (w)b$ is a tensor product of a with b parametrized by weights w

To better understand how the tensor product of irreps is computed, let's consider $u^{(l_1)} \in R^{2l_1+1}$ and $v^{(l_2)} \in R^{2l_2+1}$. Then $u^{(l_1)} \otimes v^{(l_2)} = (u \otimes v)^{(l)} \in R^{2l+1}$ can be calculated using Clebsch-Gordan coefficients denoted with C:

$$(u \otimes v)^{(l)} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C_{(l_1,m_1)(l_2,m_2)}^{(l,m)} u_{m_1}^{(l_1)} v_{m_2}^{(l_2)}$$

Specifically, the tensor product between an input feature of order l_1 and a convolutional filter of order l_2 yields irreducible representations of output orders $|l_1 - l_2| \leq l \leq |l_1 + l_2|$, which in NequIP are constrained to have a maximum rotation order l_{max}. All of this functionality is conveniently implemented in 'e3nn'[4] as the 'e3nn.o3.TensorProduct' class.

After this convolution, the node features undergo another pass through a Self-Interaction layer, as described earlier. Subsequently, the updated node features might also be added to the node features from the previous block via a ResNet-style skip-connection (Figure 1B). Whether this skip-connection occurs or not is determined by a hyperparameter. Lastly, the combined features are processed by an equivariant SiLU-based gate nonlinearity[6] facilitated by 'e3nn.nn.Gate'. Scalar features are exempt from gating and are processed directly by SiLU instead.

Output Block

After a sequence of Interaction Blocks, the features corresponding to $l = 0$ from the final convolution are directed to an Output Block (Figure 1A). This Output Block comprises two Atomwise Linear self-interaction layers that yield atomic energies $E_{i,atomic}$, which are then aggregated (summed) to output the total predicted energy of the system.

Dataset and Data Featurization

In the dataset under investigation, the predicted value is also a single scalar, but it denotes the area under the concentration-time curve (AUC), a pharmacokinetic metric quantifying the total exposure to a drug over time.

The primary distinction of my model from NequIP lies in its input structure. While NequIP solely relies on atomic numbers as input, my model incorporates additional information, including slightly different atom types and partial atomic charges. These atom types were assigned based on the General Amber Force Field (GAFF)[7] parametrization by a program called ACPYPE.[8] Unlike atomic numbers, these atom types encode details about chemical bonds and the chemical environment of each atom. A selection of these atom types with their definitions in GAFF is provided in Table 1. Across all ligands in my dataset, the total number of these atom types was 22 compared to the 7 atomic numbers. The partial charges on atoms were determined using the AM1-BCC charge scheme, also using ACPYPE.

No.	Atom type	Description	No.	Atom type	Description
1	c	sp^2 carbon in C═O, C═S	2	c1	sp^1 carbon
3	c2	sp^2 carbon, aliphatic	4	c3	sp^3 carbon
5	ca	sp^2 carbon, aromatic	6	n	sp^2 nitrogen in amides
7	n1	sp^1 nitrogen	8	n2	sp^2 nitrogen with 2 subst., real double bonds
9	n3	sp^3 nitrogen with 3 subst.	10	n4	sp^3 nitrogen with 4 subst.
11	na	sp^2 nitrogen with 3 subst.	12	nh	amine nitrogen connected to aromatic rings
13	no	Nitrogen in nitro groups	14	o	sp^2 oxygen in C═O, COO^-
15	oh	sp^3 oxygen in hydroxyl groups	16	os	sp^3 oxygen in ethers and esters
17	s2	sp^2 sulfur (p═S, C═S, etc.)	18	sh	sp^3 sulfur in thiol groups
19	ss	sp^3 sulfur in —SR and S—S	20	s4	hypervalent sulfur, 3 subst.

Table 1: GAFF atom types and their definitions.

After preprocessing the ligands and obtaining the new atom types and partial charges, the atom types were one-hot encoded as before and then concatenated with the partial charges to form node features to be passed through the first Atomwise Linear layer in the Embedding Block (Figure 2). Additionally, a hyperparameter optimization was conducted over the parameters listed in Table 2 to determine the setup that yields the highest accuracy.

3 Results

Hyperparameter Optimization

In the first experiment, the hyperparameters that achieve the highest accuracy for NequIP with atomic numbers as input were determined through model training and validation. This process involved plotting train/validation loss curves and comparing R2-scores between the predicted and true values of AUC, which served as a metric for the model's accuracy. The train/validation loss curves can be found in Appendix A, while a summary of the hyperparameter optimization along with the corresponding R2-scores is provided in Table 2. Hyperparameter values highlighted in cyan were subsequently utilized in further experiments.

HP Name	HP Value	R^2 score	HP Name	HP Value	R^2 score	HP Name	HP Value	R^2 score
LMAX	0	0.63	INVARIANT_LAYERS	1	0.64	RESNET	True	0.59
	1	0.65		2	0.65		False	0.64
	2	0.67		3	0.62	USE_SC	True	0.63
	3	0.63	INVARIANT_NEURONS	16	0.60		False	0.49
	4	0.65		32	0.64	BATCH_SIZE	2	0.56
NUM_BASIS	6	0.62		64	0.64		5	0.65
	8	0.63		128	0.63		10	0.65
	10	0.58	NUM_CONV_LAYERS	1	0.62	LEARNING_RATE	1e-2	0.46
BNI	8	0.71		2	0.64		5e-3	0.52
	16	0.67		3	0.66		1e-3	0.66
	32	0.60		4	0.70		1e-4	0.52
	64	0.57		5	0.69		1e-5	0.52

Table 2: A summary of the hyperparameter optimization results. Parameters associated with the highest accuracy are highlighted in cyan. The table includes the parameters specified below.

- LMAX: The maximum rotation order.
- NUM_BASIS: The number of Bessel Radial Bases.
- BNI: Base number of irreps.
- INVARIANT_LAYERS: The number of linear layers in the $R(r_{ij})$ MLP.
- INVARIANT_NEURONS: The number of hidden neurons in the $R(r_{ij})$ MLP.
- NUM_CONV_LAYERS: The number of sequential Interaction Blocks.
- RESNET/USE_SC: Indicates whether to apply skip-connection or not.

The hyperparameter that exerted the most pronounced impact on the model's accuracy was BNI (base number of irreps), which represents the multiplicity of irreps of each type in hidden layers. For instance, if $BNI = 16$ and $l_{max} = 1$, the node feature irreps would be "16x0e" for the Embedding Block and "16x0e+16x1o" for the Convolution layer. The need for a smaller number of irreps in training the data could be attributed to the dataset's relatively small size (approximately 300 molecules) and the structural similarities among the ligands. Notably, when larger values of BNI were used, overfitting was observed, as evidenced by the train/validation curves depicted in Figure 3.

Atom Types and Partial Charges

In the second experiment, the impact of substituting atomic numbers with more diverse atom types as the model's input, along with augmenting node features with partial atomic charges as described in the Methods section, was investigated. The outcomes of these experiments are summarized in Table 3.

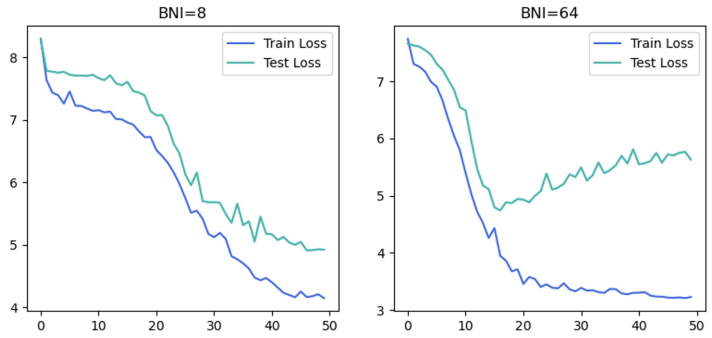

Figure 3: Plotted train/validation loss curves for $BNI = 8$ and 64. The plot for the larger value (on the right) indicates overfitting occurring after around epoch 20.

Featurization	**R^2-score, average**
Atomic Numbers	0.70
GAFF Atom Types	0.76
GAFF Atom Types + Charges	0.79

Table 3: Average R2-scores for the three models.

Both modifications yielded models with higher accuracies. This improvement could be attributed to the fact that the new atom types inherently capture more chemically relevant information, especially considering that GAFF is commonly used to parametrize small molecules for protein-ligand systems in molecular dynamics. As a result, their embeddings are better separated in the feature space, resulting in more accurate predictions. For example, carbonyl and hydroxyl oxygen atoms are now embedded into different vectors, unlike the scenario where both were assigned the same vector corresponding to the general oxygen atom based on atomic number.

The inclusion of partial atomic charges additionally resulted in a model with slightly enhanced predictive ability, while also contributing to stabilizing the training process (Figure 4). Although the improvement is not notably drastic, and the resulting R2-scores could still be improved further, these two experiments suggest that additional feature engineering may indeed bolster equivariant models for predicting small molecule binding affinities. Lastly, the model's accuracy score may not be very high because the AUC cannot be interpreted as easily as a sum of individual atom predictions like energy.

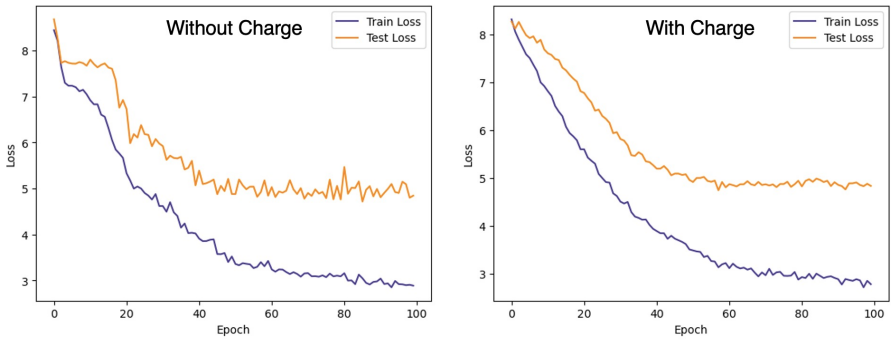

Figure 4: Train/Validation Loss Curves for the model that uses partial charges as features (on the left) and the one that doesn't (on the right).

4 Conclusion

In this study, I explored the effectiveness of equivariant neural network architectures (specifically NequIP) in predicting small molecule binding affinity. Through experimentation, I identified how hyperparameters, atom types, and partial atomic charges influence the model's performance. Hyperparameter optimization highlighted that a smaller multiplicity of irreps (BNI) leads to better generalization on a smaller dataset. The introduction of more diverse atom types and inclusion of partial atomic charges slightly improved predictive accuracy, attributed to their ability to capture finer-grained chemical information. Despite modest R2-scores, particularly in predicting the area under the concentration-time curve (AUC), these models serve as my starting point for further exploration and refinement of symmetry-aware models for small molecule binding affinity prediction in drug discovery.

References

[1] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv*, February 11 2023.

[2] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.e13, February 20 2020.

[3] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, May 4 2022.

[4] Mario Geiger and Tess Smidt. E3nn: Euclidean neural networks. *arXiv*, July 18 2022.

[5] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv*, May 18 2018.

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv*, June 5 2023.

[7] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. Development and testing of a general amber force field. *Journal of Computational Chemistry*, 25(9):1157–74, July 15 2004.

[8] Luciano Kagami, Alan Wilter, Adrian Diaz, and Wim Vranken. The acpype web server for small-molecule md topology generation. *Bioinformatics*, 39(6):btad350, June 1 2023.

5 Appendix

Train/Test Loss Curves For Hyperparameter Optimization

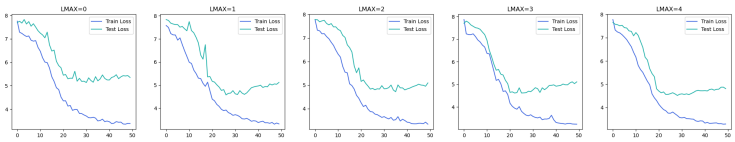

Figure 5: Train/Test Loss Curve for Hyperparameter Set 1

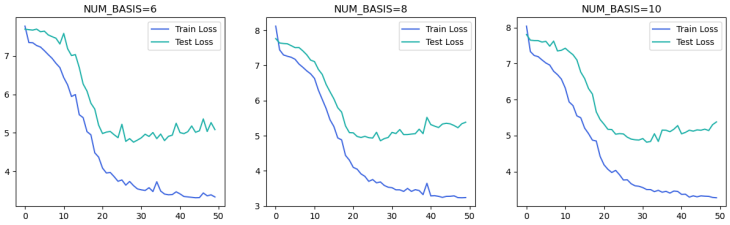

Figure 6: Train/Test Loss Curve for Hyperparameter Set 2

Figure 7: Train/Test Loss Curve for Hyperparameter Set 3

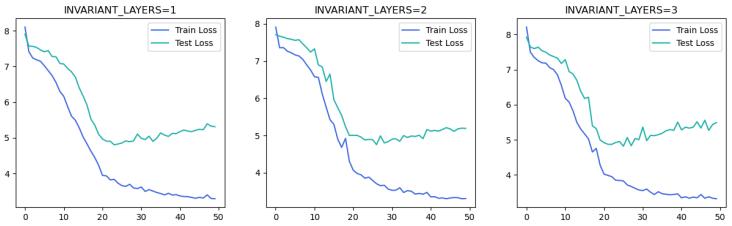

Figure 8: Train/Test Loss Curve for Hyperparameter Set 4

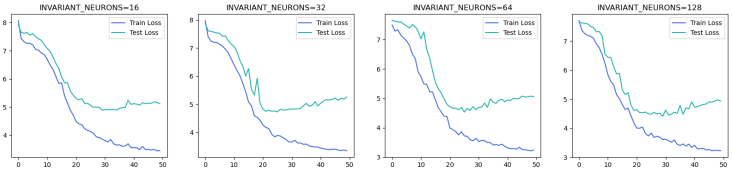

Figure 9: Train/Test Loss Curve for Hyperparameter Set 5

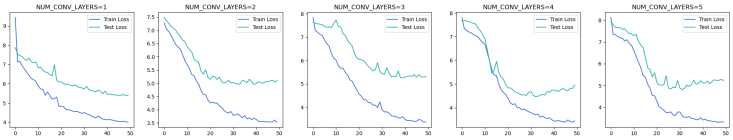

Figure 10: Train/Test Loss Curve for Hyperparameter Set 6

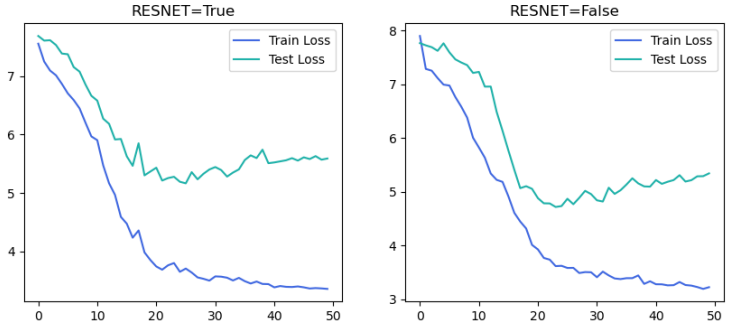

Figure 11: Train/Test Loss Curve for Hyperparameter Set 7

Figure 12: Train/Test Loss Curve for Hyperparameter Set 8

Figure 13: Train/Test Loss Curve for Hyperparameter Set 9

Figure 14: Train/Test Loss Curve for Hyperparameter Set 10