02456 Deep Learning - Final Project

Reliable Uncertainty Estimation for Neural Networks with Conformal Prediction

Group 51: My Lan Nguyen (S251583), Saffron Salmah Yen Lim (S252035), Thorri Elis Halldoruson (S252362), Fatima Nowshad (S252036)

ABSTRACT

This paper investigates reliable uncertainty estimation for image classification using conformal prediction on the CIFAR-10 dataset. We implement standard split conformal prediction on a fine-tuned ResNet-18 model and evaluate two smoothing methods: cluster-based and k-nearest-neighbors (k-NN) smoothing to evaluate whether incorporating feature-space structure leads to better efficiency. Experiments demonstrate that both smoothing approaches improve efficiency (reducing average prediction set size from 1.14 to 1.13) while maintaining coverage near the target 90% level.

1. INTRODUCTION

1.1. Background Information

As Machine Learning and Deep Learning models become more advanced, the use of Artificial Intelligence (AI) has been exceedingly integrated into daily life, including deployments in high-risk settings like in the medical, defense, and industrial sectors, amongst others. Despite huge progress in the field, AI methods are not infallible and there is always the possibility of model failure, in which untrue or even harmful outputs may be produced [1]. This is especially so in the case of black-box models, whose inherent complexity limits a user's understanding of its internal decision-making processes and reduces the interpretability of the model output [2], in turn diminishing user trust and model utility. Uncertainty Quantification (UQ) estimates the reliability or statistical confidence in a model's predictions for new or unseen data points [3]. As models tend to be overconfident and uncalibrated when it comes to estimating their own uncertainty, additional methods may be required [4].

Conformal Prediction (CP) provides distribution-free, finite-sample-calibrated prediction sets for machine learning classifiers. As CP is distribution-free, there is no need for any underlying assumptions about the data to be met, such as normality. Additionally, CP is model-agnostic, meaning that it can serve as a "post-hoc wrapper" around any model without necessitating retraining or making any changes to the model's internal structure. Given any base predictor, CP constructs a prediction set $T(x)$ that contains the true label with probability at least $1 - \alpha$, for any user-specified $\alpha \in (0, 1)$. For example, if a user requests 95% coverage ($\alpha = 0.05$), the algorithm mathematically guarantees that the true label will be contained in the output set 95% of the time in expectation [1]. While CP provides reliable uncertainty quantification, a key challenge remains: prediction sets may be unnecessarily large, especially for difficult examples or when feature space geometry is ignored. This motivates methods that use representation structure, such as clustering or neighborhood information, to refine nonconformity scores.

1.2. Prior Work

Early work on conformal prediction established its core guarantee: given a model and a calibration set, prediction sets can be constructed with finite-sample, distribution-free coverage. Angelopoulos and Bates (2023) [1] provide the modern, unified treatment of these methods, outlining both their theoretical guarantees and practical use across classification, regression, and structured prediction. Their analysis highlights an important limitation of standard split conformal prediction: while coverage is guaranteed, the resulting sets may be overly large when the nonconformity scores fail to reflect underlying structure in the feature space. A growing line of research addresses this efficiency issue by incorporating representation geometry into the nonconformity scores. Zargarbashi and Avestimehr (2023) demonstrate this in the context of graph neural networks, where local neighborhood information—obtained via k-nearest neighbors or graph-aware similarity—significantly tightens the prediction sets without compromising coverage [4]. Their results show that leveraging latent-space structure can meaningfully improve conformal efficiency, particularly on complex, high-dimensional data.

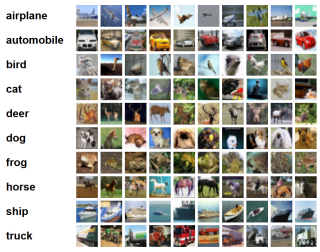

Figure 1: Samples from the CIFAR-10 classes [5].

Our work is motivated directly by these observations. We extend the idea of geometry-aware conformal prediction to the image domain by applying smoothing mechanisms—cluster-based averaging and k-NN propagation—to ResNet-18 latent representations. This allows us to evaluate whether the benefits observed in graph domains transfer to vision tasks, and whether local feature-space structure can yield tighter prediction sets while preserving the target 90% marginal coverage.

1.3. Project Scope & Objectives

In this project, the following methods are explored:

- Baseline Split Conformal Prediction on the CIFAR-10 dataset using a pretrained ResNet-18 classifier.
- Cluster-based averaging, where the nonconformity score is regularized by the mean nonconformity score of its corresponding cluster.
- k-NN–based smoothing, which propagates nonconformity via local neighborhoods in feature space.

Our objective is to evaluate whether incorporating feature-space structure leads to improved efficiency (smaller prediction set size) while maintaining coverage close to the target 1 - α = 90%.

1.4. Project Code

The code and data used is fully accessible at this link.

2. METHODS

2.1. Data

We use the CIFAR-10 dataset which consists of 50,000 training images and 10,000 test images, with a total of 10 balanced classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) (Figure 1). All images are resized to 224×224 (to fit ResNet-18), normalized, and augmented with horizontal flips for training [6].

The 50,000 training images were split following a 60-20-20 split for the training set, calibration set, and test set respectively. The training set is used for fitting the classifier; the calibration set is used to find the quantile; and finally the test set is used to evaluate final coverage and size.

2.2 ResNet-18

ResNet-18 is a lightweight 18-layer residual convolutional neural network designed for efficient image classification. The architecture introduces residual (skip) connections that allow gradients to flow directly across multiple layers, addressing the vanishing gradient problem and enabling effective training of deep networks. ResNet-18 consists of an initial convolution layer followed by four residual stages, each containing multiple residual blocks composed of two 3×3 convolutional layers with batch normalization and ReLU activations, concluding with global average pooling and a fully connected classification layer [7]. We utilise the ResNet-18 model from the pytorch torchvision library, which was pre-trained on the ImageNet-1K dataset [8], before fine-tuning it on the CIFAR-10 dataset.

2.3. Baseline Conformal Prediction

Standard Split Conformal Prediction [1] serves as the baseline method for this project. For the baseline, we fine-tune a pretrained ResNet-18 on the CIFAR-10 train split. For each sample in the calibration set, the nonconformity score s_i is:

$$s_i = 1 - softmax(z_y)$$

Where z_y is the logit for the true class y. We take the $(1 - \alpha)$ empirical quantile of these scores to obtain the threshold $\hat{q}$. At evaluation time, an image receives a prediction set containing all classes whose softmax probability is at least $1 - \hat{q}$. This serves as the reference point for the smoothing methods. In this project, α is set to a value of 0.1, i.e., the probability that the prediction set contains the correct label is guaranteed to be at least 90% in expectation.

2.4. Cluster-based Smoothing

We embed the calibration images using the ResNet-18 latent features before the final fully connected layer and run k-means clustering to partition calibration points into K clusters. Each calibration sample has:

- its own score s_i
- the average score of its cluster c:

$$\overline{s_c} = \frac{1}{|C_c|} \sum_{i \in C_c} s_i$$

We create a smoothed score, as adapted from [4]:

$$\tilde{s}_i = (1 - \lambda) s_i + \lambda \overline{s_C}$$

Where $\lambda \in [0, 1]$ is a diffusion parameter that controls the blending strength between the original score and the smoothed score [4], i.e., $\lambda = 0$ means that no smoothing occurs, while $\lambda = 0.5$ means that there is equal weight on the

sample's own score s_i and the average cluster score. We compute $\hat{q}$ from these smoothed scores and form prediction sets using the same rule as the baseline. This method tests whether broad feature-space structure can stabilize scores and shrink prediction sets.

2.5. k-NN-based Smoothing [9]

Instead of clusters, we use local neighborhoods. Raw nonconformity scores s_i may be inherently noisy estimators of local error rates, due to potential noise in the data. Hence, kNN-smoothing addresses this by constructing a local empirical score distribution via neighborhood averaging. For each calibration sample x_i, we find its k nearest neighbors in feature space and compute the average of their scores:

$$\overline{s_{kNN}} = \frac{1}{k} \sum_{j \in kNN(x_i)} s_j$$

The smoothed score , as adapted from [4] becomes:

$$\tilde{s}_i = (1 - \lambda) s_i + \lambda \overline{s_{knn}}$$

Where $\lambda \in [0, 1]$ is a diffusion parameter that controls the blending strength [4]. We again compute $\hat{q}$ from these smoothed scores. This approach is expected to capture local geometry more precisely than clustering and may lead to more adaptive prediction sets.

2.6. Evaluation Metrics

The following evaluation metrics [4] are used to measure the effectiveness of the kNN-based smoothing and the cluster-based smoothing in improving CP compared to the baseline standard-split CP:

Coverage: Coverage measures the actual proportion of the test set where the true label is successfully captured within the predicted set. For a desired confidence level of $1-\alpha$, the coverage should be approximately equal to $1-\alpha$.

Efficiency: Efficiency refers to the average size of the prediction sets produced by the conformal method. A more efficient method produces smaller sets on average, which are more useful and informative. For example, a smaller set containing {Cat, Dog} is more efficient than a larger set containing {Cat, Dog, Fox, Wolf}.

Singleton-Hit Ratio: Singleton-Hit Ratio measures the fraction of the test set for which the prediction set contains exactly one label (a "singleton"), and that single label is the correct one.

3. RESULTS

3.1. Baseline Conformal Prediction

We first evaluate the performance of the standard split conformal prediction set-up based on the metrics we have listed. The baseline method achieves the following results on the calibration set: Coverage = 90.16%, Efficiency = 1.14, Singleton-Hit Ratio = 94.969%.

The baseline model achieves a coverage of ~90%, which is close to the target 1 - α = 90% marginal coverage. These values serve as a baseline to compare against and as a reference point for the smoothing methods.

3.2 Hyperparameter Sweeping

Before Cluster-based and kNN-based smoothing, we first conduct a hyperparameter sweep across the parameter space.

Cluster smoothing: K ∈ {50, 110, 170} and λ ∈ {0.01, 0.02, 0.05, 0.1}; where K is the number of clusters and λ is the diffusion parameter. Here, we report the performance metrics for the optimal setting of K = 110. Further details can be found in Appendix A.

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.13%	89.73%	1.14	1.13
0.02	90.06%	89.67%	1.13	1.13
0.05	89.79%	89.42%	1.13	1.12
0.1	89.56%	89.18%	1.12	1.11

Table 1: Metric values for Cluster-smoothing for hyperparameter sweeps for K=110

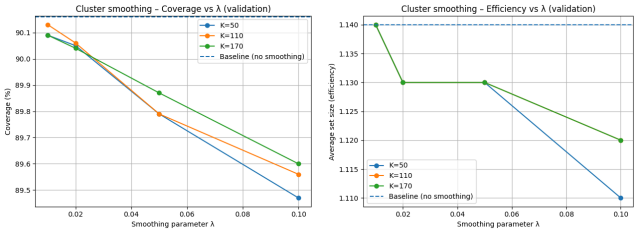

Figure 2: Coverage and Average Set Size against λ for different K

k-NN smoothing: k {2, 8, 14} and λ ∈ {0.01, 0.025, 0.05, 0.1}; where k is the number of neighbours and λ is the diffusion parameter. Here, we report the performance metrics for the optimal setting of k = 8.Further details can be found in Appendix A.

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.14%	89.74%	1.14	1.13
0.025	90.09%	89.70%	1.13	1.13
0.05	89.99%	89.57%	1.13	1.13
0.1	89.78%	89.41%	1.12	1.12

Table 2: Metric values for kNN-smoothing for hyperparameter sweeps for k=8

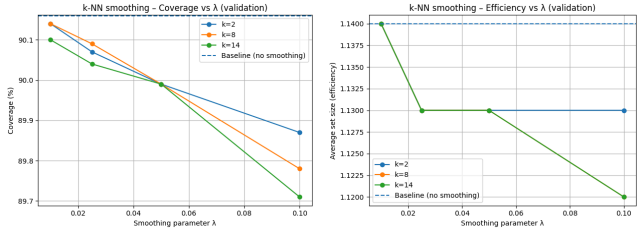

Figure 3: Coverage and Average Set Size against λ for different k

3.3 Cluster-based Smoothing

For cluster-based smoothing, we begin by setting an appropriate value for the parameters — the number of clusters, K and the diffusion parameter, λ . Setting K = 110 and λ = 0.02, we obtain: Coverage = 90.06 %, Efficiency = 1.13, Singleton-Hit Ratio = 94.973%.

Compared to the baseline, clustering slightly improves efficiency (reducing the average set size from 1.14 to 1.13) while maintaining identical coverage of above 90% and similar singleton hit ratio. This slight efficiency gain is mainly from a decrease in the average size of non-singleton sets.

3.4. k-NN-based Smoothing

For the kNN-based smoothing, by setting an appropriate value for the parameters — number of neighbours, k and the diffusion parameter, λ. Setting k = 8 and λ = 0.025, we obtain: Coverage = 90.09 %, Efficiency = 1.13, Singleton-Hit Ratio = 94.973%.

k-NN smoothing achieves nearly the same performance as clustering at these parameter settings, with a potential of superior robustness. This method again improves efficiency relative to the baseline while preserving coverage and singleton performance.

4. DISCUSSION

Our experiments demonstrate that incorporating feature-space information into conformal prediction by smoothing improves the efficiency of prediction sets while maintaining the desired coverage. Both smoothing methods successfully reduced the average prediction set size compared to the baseline standard split conformal prediction (efficiency = 1.14), with k-NN smoothing achieving an efficiency of 1.13 while maintaining valid coverage (90.09%).

A clear trade-off between efficiency and coverage stability was observed (Figures 2 & 3) . Cluster smoothing reduces variance in calibration scores but may oversmooth due to cluster mixing, reducing coverage more aggressively at higher λ. Furthermore, cluster-based smoothing proved sensitive to the choice of λ. While it aggressively reduced set sizes (down to 1.07), it frequently caused coverage to dip below the 90% target (e.g., 89.56% at λ = 0.1). This suggests that global cluster assignments may over-generalize the nonconformity of specific samples, "polluting" the nonconformity scores of difficult examples with those of easier ones in the same cluster.

k-NN smoothing, which leverages local neighborhoods, provides a more faithful representation of difficulty and better robustness compared to the cluster-based smoothing. By defining neighborhoods locally for each point, it avoided the negative side effects of sticking to fixed cluster structures. The results show that a moderate smoothing strength (λ = 0.025) with k = 8 neighbors offers the optimal balance, tightening prediction sets without violating the coverage guarantee. As a result, it achieves superior coverage–efficiency trade-offs and generalizes more reliably between calibration and test sets. Interestingly, both methods showed stable performance on new, unseen test data, suggesting that the smoothing successfully sharpens the prediction sets during the calibration phase without damaging the model's overall ability to generalize.

5. CONCLUSION

This project confirms that incorporating feature-space structure through the use of k-NN (local) and cluster-based (global) smoothing methods leads to better efficiency while maintaining coverage close to the target 90%. While standard split conformal prediction on CIFAR-10 is already highly efficient (average set size 1.14), smoothing mechanisms can further improve this. k-NN based smoothing emerged as the most effective method by leveraging local latent structure. Although cluster-based smoothing reduced set sizes more than the k-NN based smoothing, it proved sensitive to the diffusion parameter λ while causing coverage to consistently dip below 90%.

6. REFERENCES

[1] A. N. Angelopoulos and S. Bates, *"Conformal Prediction: A Gentle Introduction,"* Foundations and Trends® in Machine Learning, vol. 16, no. 4, pp. 494–591, 2023.

[2] C. Rudin, "*Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*," Nature Machine Intelligence, vol. 1, no. 5, pp. 206–215, May 2019, doi: https://doi.org/10.1038/s42256-019-0048-x.

[3] S. Seoni, V. Jahmunah, M. Salvi, P. D. Barua, F. Molinari, and U. R. Acharya, "*Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013-2023)*," Computers in Biology and Medicine, vol. 165, p. 107441, Oct. 2023, doi: https://doi.org/10.1016/j.compbiomed.2023.107441.

[4] S. H. Zargarbashi and S. Avestimehr, *"Conformal Prediction Sets for Graph Neural Networks,"* Proceedings of the 40th International Conference on Machine Learning (ICML), PMLR vol. 202, pp. 41373–41392, 2023.

[5] A. Krizhevsky, "*CIFAR-10 and CIFAR-100 datasets*," Toronto.edu, 2009. https://www.cs.toronto.edu/~kriz/cifar.html

[6] A. Krizhevsky, "*Learning Multiple Layers of Features from Tiny Images*," Apr. 2009. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[7] K. He, X. Zhang, S. Ren, and J. Sun, "*Deep Residual Learning for Image Recognition*," Dec. 2015. Available: https://arxiv.org/pdf/1512.03385

[8] "*resnet18 — Torchvision main documentation*," Pytorch.org, 2024.https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html#torchvision.models.ResNet18_Weights

[9] G. Lugosi and M. Matabuena, "*Conformal and kNN Predictive Uncertainty Quantification Algorithms in Metric Spaces*," Arxiv.org, Jul. 21, 2025. https://arxiv.org/html/2507.15741v1

DECLARATION OF USE OF GENERATIVE AI

This declaration **must be filled out** and included as the **final page** of the document. The questions apply to all parts of the work, including research, project writing, and coding.

- I/we have used generative AI tools: [~~yes~~ / **no**]

If you answered *yes*, please complete the following sections. List the generative AI tools you have used:

-

Describe how the tools were used:

What did you use the tool(s) for?

At what stage(s) of the process did you use the tool(s)?

How did you use or incorporate the generated output?

APPENDIX A

Cluster-smoothing Hyperparameter Sweeping Results:

For K = 50

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.09%	89.72%	1.14	1.13
0.02	90.05%	89.66%	1.13	1.13
0.05	89.79%	89.42%	1.13	1.12
0.1	89.47%	89.11%	1.11	1.11

Table A1: Metric values for Cluster-smoothing for hyperparameter sweeps for K = 50

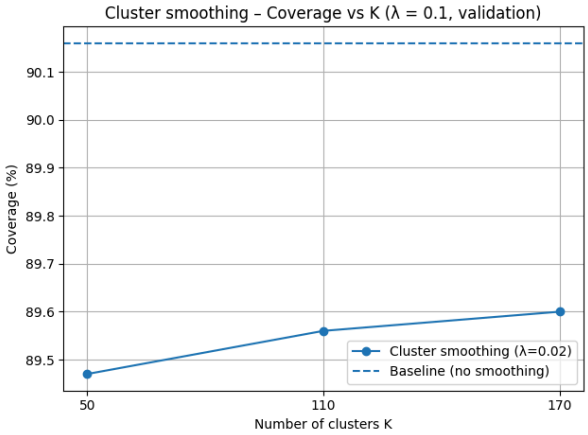

Figure A1 : Coverage vs. Number of Clusters, K for λ = 0.02

For K = 170

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.09%	89.72%	1.14	1.13
0.02	90.04%	89.66%	1.13	1.13
0.05	89.87%	89.50%	1.13	1.12
0.1	89.60%	89.19%	1.12	1.11

Table A2: Metric values for Cluster-smoothing for hyperparameter sweeps for K = 170

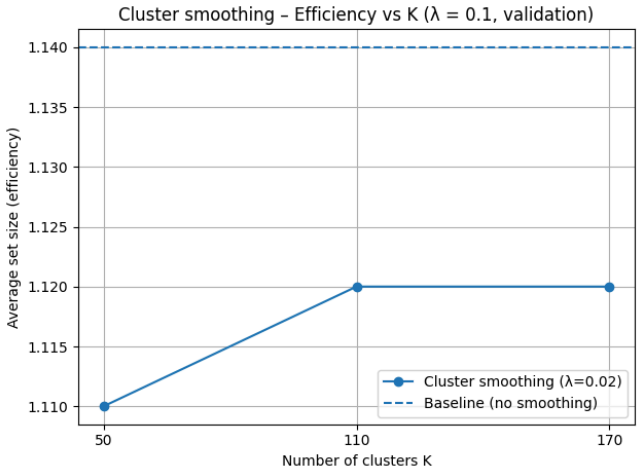

Figure A2 : Average Set Size vs. Number of Clusters, K for λ = 0.02

APPENDIX B

kNN - smoothing Hyperparameter Sweeping Results:

For k = 2

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.14%	89.76%	1.14	1.13
0.025	90.07%	89.70%	1.13	1.13
0.05	89.99%	89.55%	1.13	1.13
0.1	89.87%	89.48%	1.13	1.12

Table B1: Metric values for kNN-smoothing for hyperparameter sweeps for k = 2

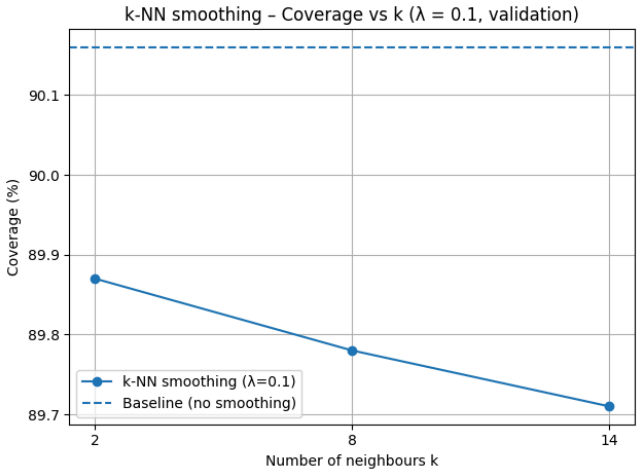

Figure B1 : Coverage vs. Number of Neighbours, k for λ = 0.1

For k = 14

λ	Coverage (Val)	Coverage (Test)	Efficiency (Val)	Efficiency (Test)
0.01	90.10%	89.72%	1.14	1.13
0.025	90.04%	89.66%	1.13	1.13
0.05	89.99%	89.57%	1.13	1.13
0.1	89.71%	89.37%	1.12	1.12

Table B2: Metric values for kNN-smoothing for hyperparameter sweeps for k = 14

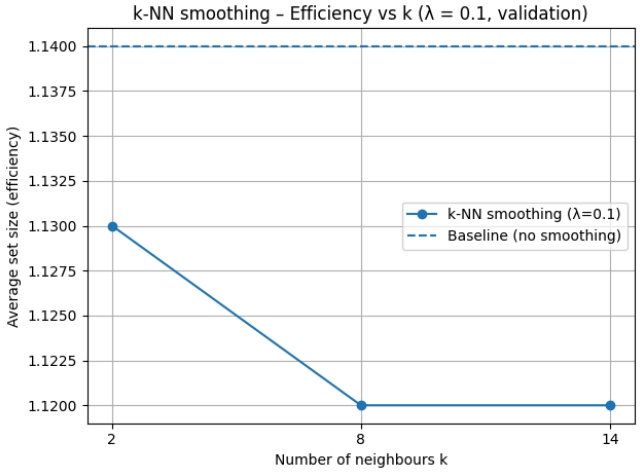

Figure B2 : Coverage vs. Number of Neighbours, k for λ = 0.1