INTEGRATING GENOMIC VARIANTS AND CLINICAL TEXT FOR CANCER TREATMENT PREDICTION: A DETERMINISTIC AND BAYESIAN DEEP LEARNING FRAMEWORK

DOWOO KIM
MENTOR: FUYUAN LYU
BRIDGE MENTORSHIP PROGRAM WINTER 2025
MCGILL UNIVERSITY COMPUTER SCIENCE

Abstract. This project investigates the use of Bayesian neural networks to quantify uncertainty in genomic variant classification, leveraging data from the MSK-IMPACT Clinical Sequencing Cohort available on Kaggle: `https://www.kaggle.com/competitions/msk-redefining-cancer-treatment/data`. The dataset includes structured genomic mutation information and unstructured clinical text, which are integrated and preprocessed to form a unified representation for predictive modeling. We first implement a deterministic deep neural network (DNN) as a baseline, evaluating performance with standard metrics such as accuracy, weighted F1-score, and ROC-AUC. To enhance model interpretability and quantify prediction confidence, we extend this framework using Monte Carlo (MC) Dropout as a Bayesian approximation, enabling the estimation of predictive uncertainty via sampling-based variance. We further assess model calibration using reliability diagrams and compute the Expected Calibration Error (ECE) for both deterministic and Bayesian settings. Through exploratory data analysis, frequency statistics, and uncertainty visualization, we identify high-variance samples, which may signal ambiguity or noisy labels. This study contributes a comprehensive methodological pipeline for cancer variant classification under uncertainty, emphasizing both predictive performance and confidence estimation. All source code, data processing, and model implementation details are available at: `https://github.com/dk1028/BRIDGE/blob/main/cancer.ipynb`.

Contents

1 Data Preprocessing and Exploratory Analysis

In our study, the initial step is to rigorously examine and preprocess the datasets, following recommended practices for biomedical data integration [Chi+18; Joh+16]. The datasets consist of structured genomic information and unstructured clinical text, which can be mathematically viewed as finite collections of samples [Ash+00].

Let

$$\mathcal{D}_{\text{variants}} = \Big\{ \big(\text{ID}_i,\ \text{Gene}_i,\ \text{Variation}_i,\ \text{Class}_i\big) \;\Big|\; i = 1, \dots, N \Big\}$$

be the training variants dataset with $N = 3321$ samples.

Similarly, let

$$\mathcal{D}_{\text{text}} = \Big\{ \big(\text{ID}_i,\ \text{Text}_i\big) \;\Big|\; i = 1, \dots, N' \Big\}$$

be the training text dataset, where $N' \approx 3321$ (with a few missing values).

1.1 Data Representation and Merging

Each sample in $\mathcal{D}_{\text{variants}}$ is a tuple containing an identifier, a categorical variable (Gene), another categorical variable (Variation), and a numerical class label. The clinical text data $\mathcal{D}_{\text{text}}$ comprises an identifier and a corresponding textual description.

A merge operation based on the common identifier ID produces the combined dataset:

$$\mathcal{D}_{\text{merged}} = \Big\{ \big(\text{ID}_i,\ \text{Gene}_i,\ \text{Variation}_i,\ \text{Class}_i,\ \text{Text}_i\big) \;\Big|\; i = 1, \dots, N \Big\}.$$

This join operation is essential to align the structured and unstructured features for subsequent modeling [AZ12]. Upon merging, we observe:

- ID, Gene, Variation, and Class are fully populated for all 3321 training samples.
- Text has a small number of missing entries (5 in the training set), prompting an imputation strategy (e.g., filling with an empty string).

1.2 Descriptive Statistics and Frequency Analysis

For the structured data, we compute standard descriptive statistics:

Mean and Standard Deviation: The sample mean $\bar{x}$ of a numerical feature (e.g., the class label) is defined as

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

and the sample standard deviation σ is computed as

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}.$$

The minimum and maximum values provide the range of the data. Such descriptive statistics help quantify central tendency and dispersion across samples [Chi+18].
Categorical Frequency Distribution: For a categorical feature such as Gene, we define the frequency function

$$f(g) = \sum_{i=1}^{N} \mathbf{1}_{\{\text{Gene}_i = g\}}, \quad g \in \mathcal{G}.$$

By sorting $f(g)$ in descending order, we identify the top 10 most frequent genes. Mathematically, this is equivalent to ranking elements of the multiset $\{\text{Gene}_1, \text{Gene}_2, \ldots, \text{Gene}_N\}$ by frequency. Figure 1 illustrates this distribution as a bar chart. The x-axis lists the top 10 genes, while the y-axis represents the count $f(g)$. Notably, BRCA1 appears most frequently, indicating potential importance in subsequent classification.

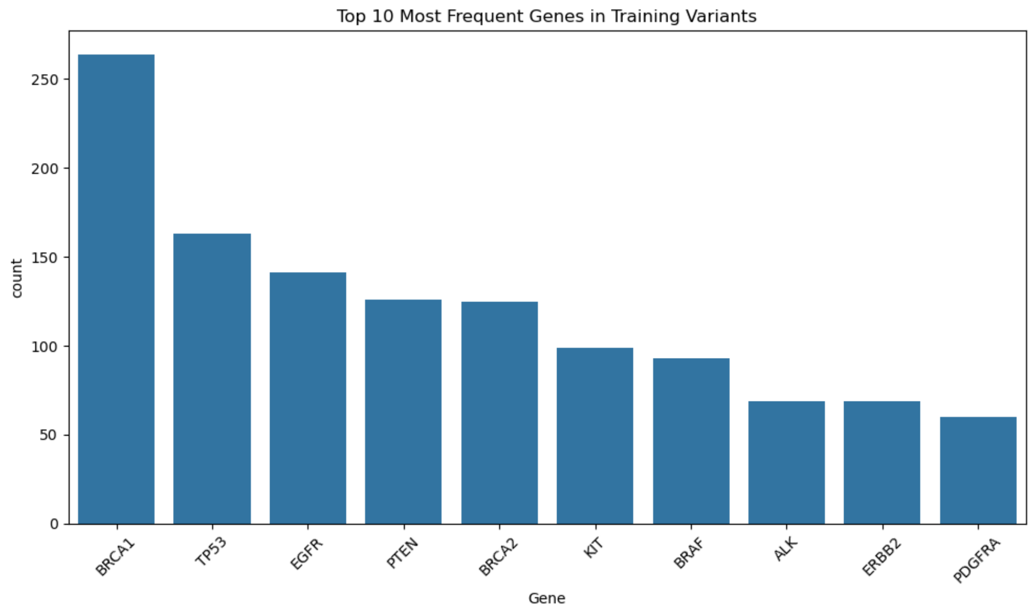

FIGURE 1. Top 10 Most Frequent Genes in the Training Variants.

1.3 Analysis of Missing Data

We inspect each column to count missing entries. For a given column C, define the indicator function

$$\mathbf{1}_{\{C_i \text{ is missing}\}},$$

and compute

$$\text{Missing}(C) \;=\; \sum_{i=1}^{N} \mathbf{1}_{\{C_i \text{ is missing}\}}.$$

This quantifies data quality and guides imputation strategies. Empirically, the structured fields have zero missing values, whereas Text exhibits 5 missing entries out of 3321. Similarly, the test data has only 1 missing text entry among 5668 samples, suggesting high overall data integrity [Joh+16].

1.4 Textual Data and Length Distribution

Each clinical text entry Text_i is a sequence of characters. We define the random variable L_i as the length (number of characters) of the text:

$$L_i \;=\; \text{length}(\text{Text}_i).$$

By computing L_i for all i, we obtain a sample distribution of text lengths. A histogram, combined with a kernel density estimate (KDE), approximates the probability density function $p_L(\ell)$ of text length:

$$\hat{p}_L(\ell) \;=\; \frac{1}{Nh} \sum_{i=1}^{N} K\Big(\tfrac{\ell - L_i}{h}\Big),$$

where K is a chosen kernel (often Gaussian) and h is the bandwidth. Classic references on KDE are provided in [Sil86].

Figure 2 displays the distribution of clinical text lengths. The x-axis represents the number of characters, and the y-axis shows the count (or density). The distribution is right-skewed, indicating that most samples have relatively short text, but a minority extend to higher character counts (over 100,000 characters). This insight informs text-processing strategies (e.g., truncation or hierarchical modeling of very long documents) [AZ12].

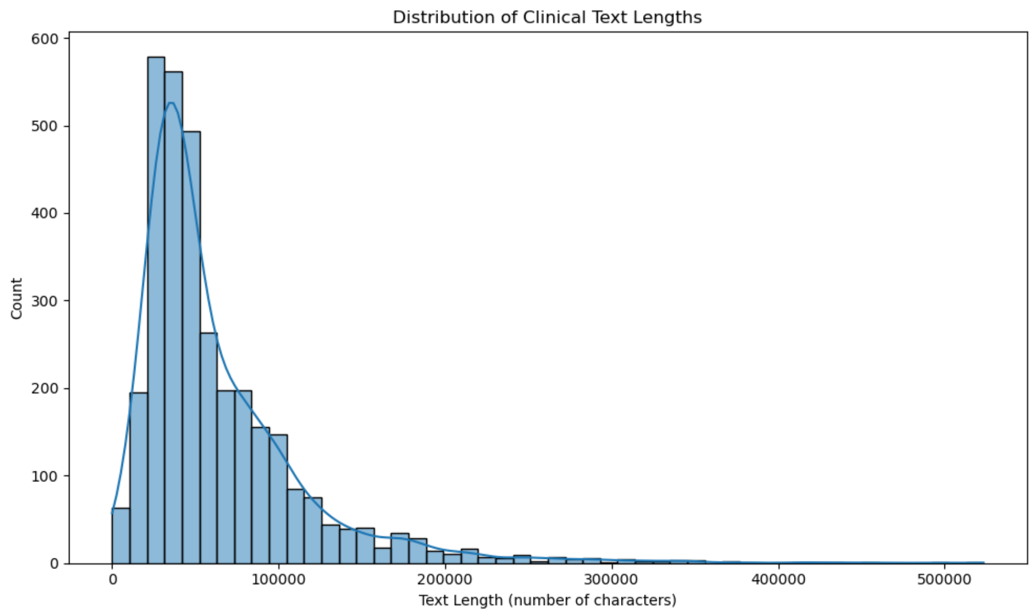

FIGURE 2. Distribution of Clinical Text Lengths in the Training Set.

1.5 Variant Classification Exploration

Although the dataset does not include a `Variant_Classification` column, the exploratory code checks if such a column exists. In general, for any categorical variable C, the frequency distribution $f(c)$ for $c \in \text{Categories}(C)$ provides insights into data composition and potential class imbalances [Chi+18]. Here, no additional variant classification data was available, so no further analysis was performed on this dimension.

2 Baseline Deep Neural Network

In this section, we construct and train a baseline Deep Neural Network (DNN) on a combination of structured (encoded) and unstructured (TF-IDF-transformed) features. The goal is to classify cancer-related variants into multiple classes based on genomic and textual data [GBC16].

2.1 Categorical Encoding

We have two high-cardinality categorical variables: **Gene** and **Variation**. Let G be the set of unique gene labels and V be the set of unique variation labels in the training dataset [Ped+11]. We define:

$$\text{Gene\_Encoded}_i = \ell_{\text{gene}}(\text{Gene}_i), \quad \text{Variation\_Encoded}_i = \ell_{\text{variation}}(\text{Variation}_i),$$

where $\ell_{\text{gene}} : G \to \{0, 1, \ldots, |G| - 1\}$ and $\ell_{\text{variation}} : V \to \{0, 1, \ldots, |V| - 1\}$ are label encoding functions. These map each categorical label to a unique integer, enabling them to be used as numeric inputs [Ped+11].

2.2 Text Vectorization via TF-IDF

We represent clinical text Text_i as a vector in $\mathbb{R}^M$ using the Term Frequency–Inverse Document Frequency (TF-IDF) transformation [Ram03]. For a given vocabulary

$$\{w_1, w_2, \ldots, w_M\}$$

of size M (e.g., $M = 5000$ most frequent terms), each document

$$d_i = \text{Text}_i$$

is converted to a vector

$$x_i(\text{tfidf}) = \big(\text{tfidf}_{i1}, \text{tfidf}_{i2}, \ldots, \text{tfidf}_{iM}\big) \in \mathbb{R}^M,$$

where

$$\text{tfidf}_{ij} = \text{tf}(w_j, d_i) \times \text{idf}(w_j).$$

In particular,

$$\text{idf}(w_j) = \log\big(\tfrac{N}{\#\{\,d : w_j \in d\,\}}\big)$$

measures how rare w_j is across the corpus of N documents [SB88].

2.3 Standardization of Numeric Features

After label encoding, the numeric features Gene_Encoded and Variation_Encoded are standardized. For a feature vector $x \in \mathbb{R}^n$, the standardized feature z is:

$$z_i = \frac{x_i - \bar{x}}{\sigma},$$

where $\bar{x}$ is the empirical mean of the feature and σ is its empirical standard deviation. This ensures that each numeric feature has zero mean and unit variance, often stabilizing neural network training [GBC16].

2.4 Feature Concatenation and Label Preparation

We concatenate the standardized numeric features z_i and the TF-IDF vector $x_i(\text{tfidf})$ for each sample i:

$$x_i = \big[\, z_i \,\|\, x_i(\text{tfidf}) \big] \;\in\; \mathbb{R}^{m+M},$$

where m is the dimension of the standardized numeric features (here $m = 2$) and M is the size of the TF-IDF vocabulary (e.g., $M = 5000$). This yields a final feature matrix

$$X \;=\; \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_N^\top \end{bmatrix} \;\in\; \mathbb{R}^{N\times(m+M)}.$$

We also shift the class labels $y_i \in \{1, \ldots, K\}$ to a zero-based index $y_i - 1$, denoting the final labels as $y_i' \in \{0, \ldots, K-1\}$. This step aligns with the sparse categorical cross-entropy loss requirement in many deep learning frameworks.

2.5 Train–Validation Split

A split is performed to partition the data into training and validation sets:

$$(X_{\text{train}}, y_{\text{train}}), \quad (X_{\text{val}}, y_{\text{val}}).$$

We use a stratified approach with test_size = 0.2, ensuring that each class is proportionally represented in both training and validation subsets [Ped+11]. Stratification preserves the label distribution $\{y_i'\}$ across subsets, reducing sampling bias.

2.6 Baseline DNN Architecture

We define a feed-forward neural network $\phi(\cdot;\theta)$ with multiple dense layers:

$$\phi(x) = \text{softmax}\Big(W^{(L)}\, \sigma\big(\cdots \sigma\big(W^{(1)}x + b^{(1)}\big)\cdots\big) \;+\; b^{(L)}\Big),$$

where:

- $W^{(\ell)}$ and $b^{(\ell)}$ are trainable weights and biases at layer ℓ,
- $\sigma(\cdot)$ is the ReLU activation,

- softmax ensures the output is a probability vector in $\mathbb{R}^K$.

We also apply dropout at various layers with probability p (e.g., 0.3 or 0.2) to mitigate overfitting [Sri+14]. Dropout randomly zeroes out a proportion of the layer's units, effectively sampling from an ensemble of sub-networks.

2.7 Training Procedure

The model is compiled using the Adam optimizer [KB15] and the sparse categorical cross-entropy loss:

$$\min_{\theta} \; -\frac{1}{N_{\text{train}}} \sum_{(x,y')\in\text{Train}} \log \phi(x;\theta)_{y'}.$$

During each epoch, the optimizer updates θ via stochastic gradient descent with momentum [Sut+13]. We track training and validation loss, as well as accuracy:

$$\text{Accuracy} = \frac{1}{N_{\text{val}}} \sum_{(x,y')\in\text{Val}} \mathbf{1}\Big(\arg\max \phi(x;\theta) = y'\Big).$$

After 10 epochs, the model converges to a final set of weights θ^*, as evidenced by the reported accuracy and loss values.

2.8 Evaluation Metrics

We compute three key metrics:

Accuracy:

$$\text{Accuracy} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \mathbf{1}\big(\hat{y}_i = y'_i\big),$$

where $\hat{y}_i = \arg\max \phi(x_i)$ is the predicted class.

Weighted F1-Score:

$$F_1 = \sum_{k=0}^{K-1} \omega_k \times F_1(k),$$

where $F_1(k)$ is the F1-score for class k, and ω_k is the proportion of samples belonging to class k. This accounts for class imbalance [Pow11].

ROC-AUC (One-vs-Rest):

$$\text{ROC-AUC} = \frac{1}{K}\sum_{k=0}^{K-1}\int_0^1 \text{TPR}_k\Big(\text{FPR}_k^{-1}(t)\Big)\,dt,$$

where TPR_k and FPR_k denote the true positive rate and false positive rate for class k. The final ROC-AUC is averaged across classes in a one-vs-rest scheme [Faw06].

2.9 Misclassification Analysis

We identify misclassified samples by comparing the predicted class $\hat{y}_i$ to the true class y'_i. The set of misclassified indices is

$$\mathcal{M} = \Big\{ i \mid \hat{y}_i \neq y'_i \Big\}.$$

The total count $|\mathcal{M}|$ quantifies how many samples the model failed to classify correctly. Such analysis can guide error inspection and data-driven model improvements.

2.10 Summary of Results

The training logs show the model progressively improving in both accuracy and loss over 10 epochs. Final metrics (accuracy ≈ 0.6195, weighted F1-score ≈ 0.6133, and ROC-AUC ≈ 0.8699) demonstrate reasonable discriminative power, although there is room for further optimization. A total of 253 misclassified samples indicates that the model could benefit from additional regularization, feature engineering, or hyperparameter tuning.

Overall, this baseline DNN approach establishes a reference point for more advanced techniques (e.g., Bayesian neural networks) by illustrating how structured and unstructured data can be combined into a unified model [GBC16]. The mathematical formulations—from label encoding and TF-IDF to standardization and neural network training—lay the groundwork for subsequent enhancements and uncertainty quantification.

3 Bayesian Approximation via Monte Carlo Dropout

In a conventional deterministic neural network, the weights remain fixed after training, producing point estimates for each input. By contrast, a Bayesian neural network aims to learn a posterior distribution over weights,

$$p(\mathbf{w} \mid \mathcal{D}),$$

where $\mathcal{D}$ denotes the training dataset [Blu+15]. Exact Bayesian inference in high-dimensional neural networks is typically intractable. Therefore, *Monte Carlo (MC) Dropout* offers a practical approximation [GG16].

3.1 Model Architecture and Training.

We define a feed-forward network

$$f(\mathbf{x}; \mathbf{w}) = \sigma_{\text{softmax}}\Big(\mathbf{W}^{(L)}\, \sigma\big(\cdots \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \cdots\big) + \mathbf{b}^{(L)}\Big),$$

where each layer may include dropout. Let $\mathbf{w}$ represent all trainable parameters (weights and biases). During training, dropout randomly zeroes out neurons with probability p (e.g., $p = 0.5$), effectively sampling from a family of sub-networks. The loss function is the standard categorical cross-entropy:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \sum_{k=1}^{K} y_{ik}\, \log f_k(\mathbf{x}_i; \mathbf{w}),$$

where $y_{ik} \in \{0, 1\}$ is the one-hot label for sample i and class k, and $f_k(\mathbf{x}_i; \mathbf{w})$ is the softmax probability for class k [GBC16].

3.2 Inference with MC Dropout.

At inference, we perform T stochastic forward passes with dropout *still active*, each time sampling a different dropout mask. Denote these forward passes by

$$\hat{\mathbf{w}}_1,\ \hat{\mathbf{w}}_2, \ldots, \hat{\mathbf{w}}_T,$$

which represent different sampled subsets of the network's neurons. The predictive distribution for an input $\mathbf{x}$ is approximated by averaging over these forward passes [GG16]:

$$\widehat{p}(y \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} f\big(\mathbf{x}; \hat{\mathbf{w}}_t\big).$$

Consequently, the mean prediction across these T samples provides a *Bayesian* estimate of class probabilities, and the sample variance quantifies *epistemic* (model) uncertainty [Blu+15].

3.3 Predictive Uncertainty and Variance.

Let $\hat{p}_t(y \mid \mathbf{x}) = f\big(\mathbf{x}; \hat{\mathbf{w}}_t\big)$ be the class-probability vector from the t-th pass. The final prediction is taken as

$$\bar{p}(y \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \hat{p}_t(y \mid \mathbf{x}).$$

For each sample i, we identify the predicted class $\hat{c}_i = \arg\max_y \bar{p}_i(y)$. The *predictive uncertainty* for that class is given by the variance:

$$\mathrm{Var}_i = \frac{1}{T} \sum_{t=1}^{T} \Big(\hat{p}_t(\hat{c}_i \mid \mathbf{x}_i) - \bar{p}_i(\hat{c}_i)\Big)^2.$$

Hence, if the network's forward passes produce widely different probabilities for the same class, the variance grows, signaling that the model is uncertain about its prediction.

3.4 Empirical Analysis and Visualization.

After training for several epochs, we evaluate the model by performing MC Dropout on the validation set. We plot a histogram of Var_i for all validation samples i. Figure 3 illustrates the distribution of predictive variances:

- **X-axis:** Predictive variance, Var_i.
- **Y-axis:** The count of validation samples within each variance bin.

A majority of samples exhibit low variance (high confidence), while a minority have significantly higher variance (low confidence).

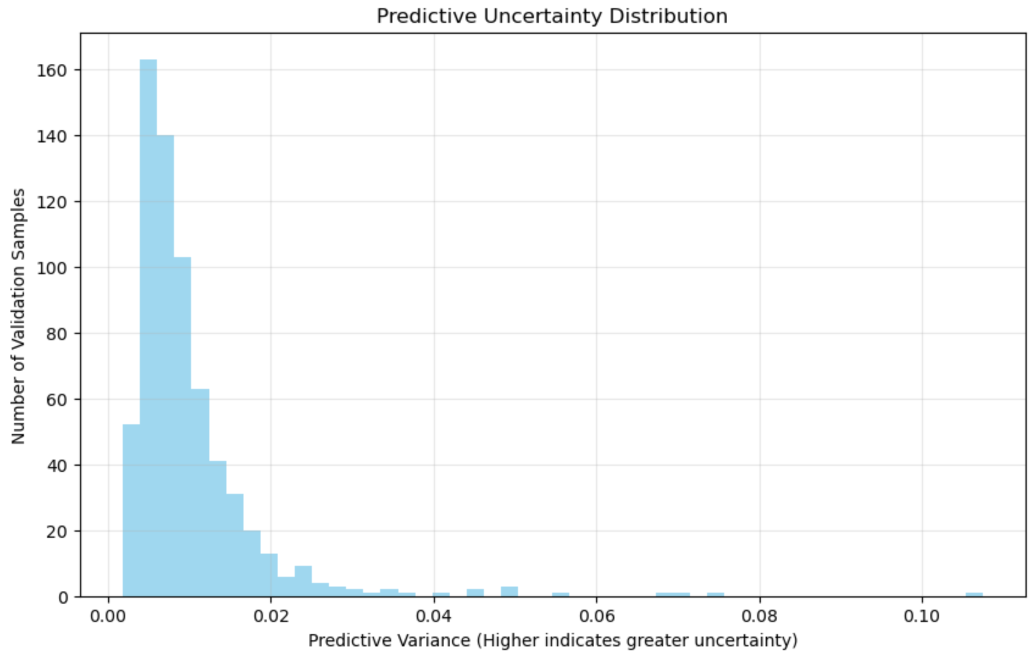

FIGURE 3. Predictive Uncertainty Distribution via MC Dropout. Higher variance indicates greater uncertainty.

To further highlight uncertain cases, we define a threshold (e.g., the 90th percentile of the variance distribution). Samples whose variance exceeds this threshold are flagged as high-uncertainty:

$$\mathcal{U}_{0.90} = \left\{ i \ \middle|\ \mathrm{Var}_i \ \geq\ \tau_{0.90} \right\},$$

where $\tau_{0.90}$ is the 90th-percentile variance. Identifying these cases can guide *active learning* or domain-expert review, focusing efforts on samples the model finds ambiguous [Set09].

3.5 Performance Metrics.

As with deterministic networks, we compute accuracy, weighted F1-score, and multi-class ROC-AUC. However, MC Dropout augments these metrics with an uncertainty measure, enabling the practitioner to:

(1) Quantify how confident the model is in each prediction.

(2) Potentially reject or manually review highly uncertain predictions.

3.6 Summary of Results.

Despite moderate accuracy (38.2%), the model provides valuable insight into predictive confidence via uncertainty estimates. Such Bayesian approximations are crucial in high-stakes domains, where decisions often require a measure of how reliable the model's predictions are [GG16; Blu+15].

4 Calibration Analysis and Uncertainty Visualization

After obtaining predictions from both the deterministic and Bayesian (MC Dropout) models, we conduct a comparative analysis to assess each model's predictive performance, calibration, and uncertainty characteristics [Guo+17].

4.1 Performance Metrics

We evaluate each model using three primary metrics:

- **Accuracy**:

$$\text{Accuracy} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \mathbf{1}\Big(\hat{y}_i = y_i\Big),$$

  where $\hat{y}_i$ is the predicted class for the i-th validation sample, and y_i is its true class.

- **Weighted F1-score**:

$$F_1 = \sum_{k=1}^{K} \omega_k \times F_1^{(k)},$$

  where $F_1^{(k)}$ is the F1-score computed for class k, and ω_k is the proportion of samples belonging to class k. This ensures that classes with more samples have a proportionate influence on the overall F1-score.

- **ROC-AUC (One-vs-Rest)**: For multi-class classification, we compute the Area Under the Receiver Operating Characteristic curve using a one-vs-rest approach. If $\text{TPR}_k(t)$ and $\text{FPR}_k(t)$ denote the true positive rate and false positive rate for class k at threshold t, then

$$\text{ROC-AUC} = \frac{1}{K} \sum_{k=1}^{K} \int_0^1 \text{TPR}_k(\text{FPR}_k^{-1}(u))\, du.$$

  A higher ROC-AUC indicates stronger discriminative ability [Faw06].

4.2 Expected Calibration Error (ECE)

Calibration reflects how well a model's predicted confidence aligns with the empirical probability of correctness. We discretize the confidence values $\hat{p}_i = \max_k f_k(\mathbf{x}_i)$ into B bins, and for each bin b compute:

$$\mathrm{acc}(b) = \frac{1}{|B_b|} \sum_{i \in B_b} \mathbf{1}\Big(\arg\max f(\mathbf{x}_i) = y_i\Big), \quad \mathrm{conf}(b) = \frac{1}{|B_b|} \sum_{i \in B_b} \hat{p}_i,$$

where B_b is the set of indices in bin b. The *expected calibration error* is defined as

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{|B_b|}{N_{\mathrm{val}}} \big|\mathrm{acc}(b) - \mathrm{conf}(b)\big|.$$

Lower ECE indicates better calibration. In practice, we often set $B = 15$ or $B = 20$ for a balance between resolution and robustness [DF83].

4.3 Reliability Diagrams (Calibration Curves)

A reliability diagram (or calibration curve) visually compares predicted confidence to the fraction of correct predictions. We plot $\mathrm{acc}(b)$ on the y-axis against $\mathrm{conf}(b)$ on the x-axis for each bin b, along with the diagonal line $y = x$ representing perfect calibration. Figure 4 exemplifies this approach, comparing the deterministic model with the Bayesian MC Dropout model.

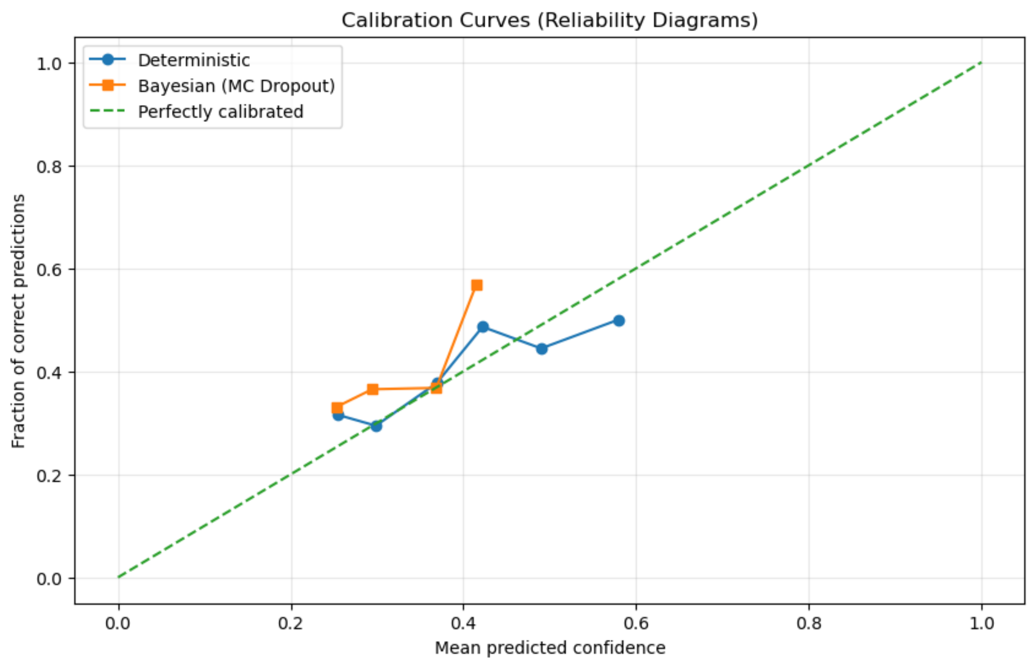

FIGURE 4. Calibration Curves (Reliability Diagrams) for Deterministic vs. Bayesian (MC Dropout) models.

4.4 Identifying High-Uncertainty Samples

In the Bayesian (MC Dropout) model, we compute an uncertainty value for each validation sample i by extracting the predictive variance for the predicted class $\hat{c}_i$. Denoting the t-th Monte Carlo forward pass as $f(\mathbf{x}_i; \hat{\mathbf{w}}_t)$, the predictive variance for class $\hat{c}_i$ is

$$\mathrm{Var}_i = \frac{1}{T} \sum_{t=1}^{T} \big(f_{\hat{c}_i}(\mathbf{x}_i; \hat{\mathbf{w}}_t) - \bar{p}_i(\hat{c}_i) \big)^2,$$

where

$$\bar{p}_i(\hat{c}_i) = \frac{1}{T} \sum_{t=1}^{T} f_{\hat{c}_i}(\mathbf{x}_i; \hat{\mathbf{w}}_t).$$

A sample is flagged as high-uncertainty if Var_i exceeds a chosen percentile threshold (e.g., the 90th percentile). Analyzing these high-uncertainty samples can guide active learning or manual review for ambiguous cases [Set09].

4.5 Uncertainty Visualizations

We use two key visualizations to interpret uncertainty:

(1) **Uncertainty Histogram:** A histogram of Var_i (Figure 5) illustrates the distribution of predictive uncertainty. The x-axis represents predictive variance, and the y-axis shows how many samples fall into each variance bin.

(2) **Scatter Plot of Uncertainty vs. Correctness:** We plot each validation sample i with coordinates (i, Var_i) (Figure 6), color-coded by whether $\hat{y}_i = y_i$ (correct) or $\hat{y}_i \neq y_i$ (incorrect). This allows us to visually inspect whether misclassifications correlate with high predictive variance.

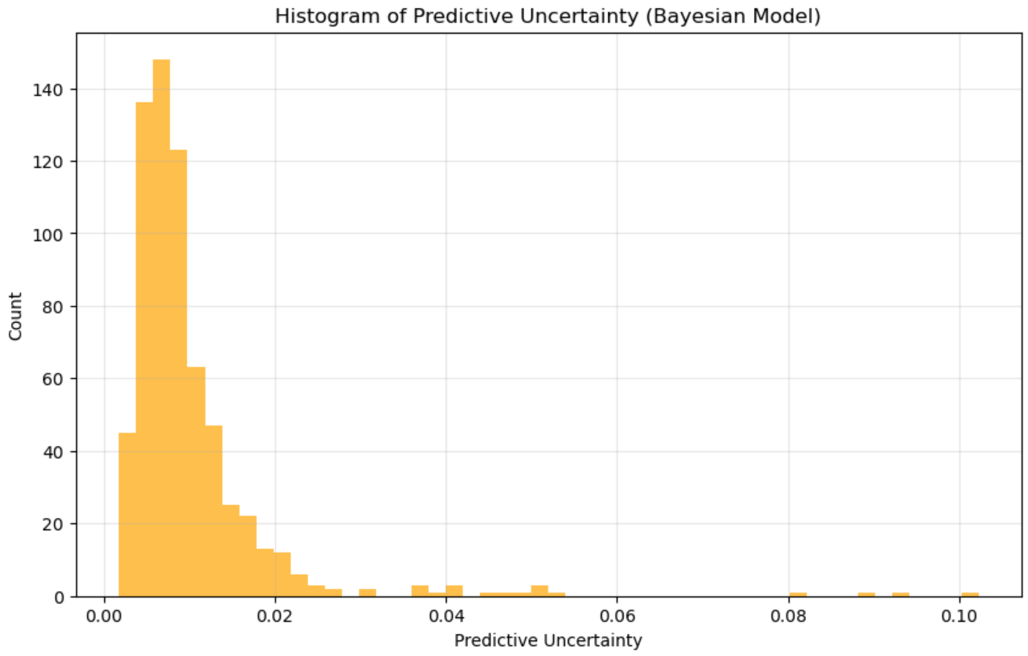

FIGURE 5. Histogram of Predictive Uncertainty (Bayesian Model). Most samples have low variance, but a minority exhibit higher variance.

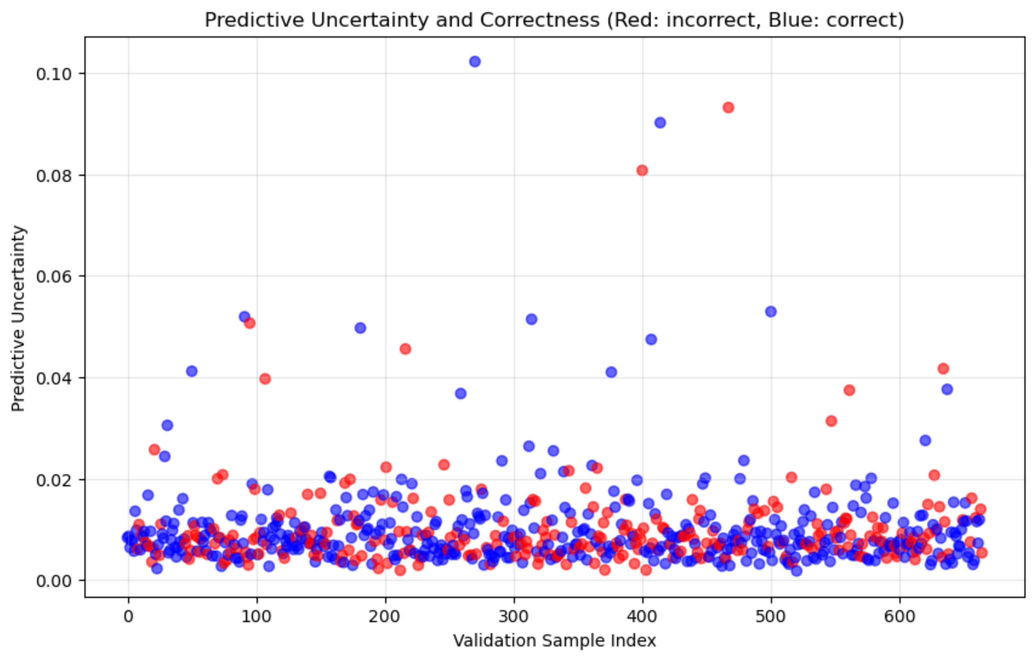

FIGURE 6. Predictive Uncertainty vs. Correctness. Blue dots are correctly classified samples, red dots are misclassified.

Interpretation. These calibration and uncertainty analyses reveal not only how accurate the models are, but also how well their predicted confidences reflect reality. The deterministic model may yield slightly lower or higher calibration error than the Bayesian model, but lacks an inherent measure of model uncertainty. In contrast, the Bayesian approach provides a quantitative estimate of confidence for each prediction, which can be invaluable in high-stakes decision-making scenarios where recognizing uncertain predictions is crucial.

5 Conclusion

In conclusion, this project demonstrates that while a deterministic deep neural network can achieve reasonably good classification performance (around 62% accuracy and a weighted F1-score above 0.60), it provides limited insight into its confidence. By contrast, a Bayesian neural network implemented via Monte Carlo Dropout yields lower accuracy (around 38%) but offers valuable uncertainty estimates, as evidenced by predictive variance and the identification of high-uncertainty samples. Calibration analyses (e.g., reliability diagrams and expected calibration error) further reveal how each model's predicted confidence aligns with true outcomes, highlighting a trade-off between raw predictive metrics and the ability to quantify uncertainty. These findings underscore the importance of selecting a modeling approach not solely based on accuracy or F1-score, but also on the capacity to flag ambiguous predictions—an essential feature in high-stakes domains like cancer treatment, where knowing when a model is unsure can be as critical as the prediction itself.

References

[Ash+00] Michael Ashburner et al. "Gene ontology: tool for the unification of biology". In: *Nature Genetics* 25.1 (2000), pp. 25–29. DOI: 10.1038/75556.

[AZ12] Charu C Aggarwal and ChengXiang Zhai. *Mining Text Data*. Springer, 2012. ISBN: 9781461432234. DOI: 10.1007/978-1-4614-3223-4.

[Blu+15] Charles Blundell et al. "Weight uncertainty in neural networks". In: *International Conference on Machine Learning (ICML)*. 2015, pp. 1613–1622.

[Chi+18] Travers Ching et al. "Opportunities and obstacles for deep learning in biology and medicine". In: *Journal of the Royal Society Interface* 15.141 (2018), p. 20170387. DOI: 10.1098/rsif.2017.0387.

[DF83] Morris H DeGroot and Stephen E Fienberg. "The comparison and evaluation of forecasters". In: *The Statistician* (1983), pp. 12–22.

[Faw06] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning (ICML)*. 2016, pp. 1050–1059.

[Guo+17] Chuan Guo et al. "On calibration of modern neural networks". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 1321–1330.

[Joh+16] Alistair EW Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific Data* 3 (2016), p. 160035. DOI: 10.1038/sdata.2016.35.

[KB15] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: (2015).

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research*. Vol. 12. 2011, pp. 2825–2830.

[Pow11] David M. Powers. "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.

[Ram03] Juan Ramos. “Using TF-IDF to determine word relevance in document queries”. In: *Proceedings of the First Instructional Conference on Machine Learning* 242 (2003), pp. 133–142.

[SB88] Gerard Salton and Christopher Buckley. “Term-weighting approaches in automatic text retrieval”. In: *Information Processing & Management* 24.5 (1988), pp. 513–523.

[Set09] Burr Settles. *Active Learning Literature Survey.* University of Wisconsin–Madison Department of Computer Sciences, 2009. URL: `http://digital.library.wisc.edu/1793/60660`.

[Sil86] Bernard W Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall/CRC, 1986. ISBN: 9780412246203.

[Sri+14] Nitish Srivastava et al. “Dropout: A simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[Sut+13] Ilya Sutskever et al. “On the importance of initialization and momentum in deep learning”. In: (2013), pp. 1139–1147.