Uncertainty Quantification in Multimodal Ensembles of Deep Learners

Katherine Brown
Tennessee Technological University
5 PUBLICATIONS **23** CITATIONS

SEE PROFILE

Farzana Ahamed Bhuiyan
Tennessee Technological University
18 PUBLICATIONS **80** CITATIONS

SEE PROFILE

Douglas A. Talbert
Tennessee Technological University
73 PUBLICATIONS **664** CITATIONS

SEE PROFILE

Uncertainty Quantification in Multimodal Ensembles of Deep Learners

Katherine E. Brown, Farzana Ahamed Bhuiyan, Douglas A. Talbert
{kebrown46, fbhuiyan42}@students.tntech.edu, dtalbert@tntech.edu
Department of Computer Science
Tennessee Technological University
1 William Jones Drive
Cookeville, TN 38505

Abstract

Uncertainty quantification in deep learning is an active area of research that examines two primary types of uncertainty in deep learning: epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty is caused by not having enough data to adequately learn. This creates volatility in the parameters and predictions and causes uncertainty. High epistemic uncertainty can indicate that the model's prediction is based on a pattern with which is it not familiar. Aleatoric uncertainty measures the uncertainty due to noise in the data. Two additional active areas of research are multimodal learning and malware analysis. Multimodal learning takes into consideration distinct expressions of features such as different representations (e.g., audio and visual data) or different sampling techniques. Multimodal learning has recently been used in malware analysis to combine multiple types of features. In this work, we present and analyze a novel technique to measure epistemic uncertainty from deep ensembles of modalities. Our results suggest that deep ensembles of modalities provide higher accuracy and lower uncertainty that the constituent single modalities and than the comparable hierarchical multimodal deep learner.

Introduction

Uncertainty in deep learning has received significant attention in recent years. There has been work in measuring uncertainty in deep neural networks (Gal and Ghahramani 2016; Kendall and Gal 2017; Lakshminarayanan and others 2017). There has also been work in utilizing uncertainty in domain-specific tasks. Uncertainty quantification has been used to identify inputs for which a human expert is needed to verify or refute an algorithm's prediction (Leibig and others 2017; Brown and Talbert 2019). Uncertainty quantification has also been applied to self-driving cars for self-detection of potentially dangerous situations (Michelmore and others 2018)

This paper uses the domain of malware detection to explore uncertainty in multimodal deep learning. Malware can cause damage to devices or leak the private information stored on them. The Android platform provides both a large pool of potential victims and ease of propagation. In May 2019, *The Verge* reports 2.5 billion active Android devices with most of these devices connected some other device at some point in their lifetime (Brandom 2019). Thus, it is important to effectively detect Android malware. Hierarchical multimodal learning has been applied to this task before, but in this work we also present an approach using ensembles of deep learners using different modalities. Additionally, we assess uncertainty quantification in both approaches.

In this work, we apply deep ensembles of modalities to malware detection. We compare these to hierarchical multimodal neural networks and neural networks from single modalities. Further, we measure and assess the associated uncertainty.

Background and Related Work

In this section, we present background and related work into uncertainty quantification in deep learning, machine learning ensembles, and multimodal learning. To our knowledge, this is the first work describing uncertainty quantification in a hierarchical multimodal neural network.

Uncertainty Quantification in Deep Learning

Uncertainty in deep learning can be divided into two categories: epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty results from an inability to model the provided data effectively (Kendall and Gal 2017). Typically, this results from not having enough training data available to allow the network to learn an effective generalization or to converge its parameters. As a result, epistemic uncertainty will be high for data that the model has not encountered and can be reduced by training the model on more data. Epistemic uncertainty is also referred to as model uncertainty since the uncertainty is inherent in the model itself (Kendall and Gal 2017). Aleatoric uncertainty, however, arises from noise in the data itself.

Epistemic uncertainty can be measured with a technique that uses active dropout layers during training and testing of the model (Gal and Ghahramani 2016). These dropout layers are placed between the weight layers of the neural network. We refer to these dropout layers as "persistent dropout." In general, dropout deactivates neurons in a specific layer with some probability (Srivastava and others 2014). In training, this provides a regularization effect. When kept active during testing, it places a distribution over the weights in the neural network (Gal and Ghahramani 2016). Measuring uncertainty of an input requires $T > 1$ forward passes through the neural network to create a probability distribution. The standard deviation of this distribution captures the degree of uncertainty associated with the input. If the predictions have a high variance or standard deviation, then the model is uncertain as to what the final prediction of the data point should be.

Ensembles of deep neural networks have been used to generate uncertainty predictions (Lakshminarayanan and others 2017). The technique consists of training an ensemble of deep neural networks to make multiple predictions on both the predicted class and the uncertainty value. Each network architecture is randomly generated to help reduce the correlation between models, and the final prediction and uncertainty value are ascertained through averaging the outputs of the model. This is combined with training the ensembles with adversarial examples to increase the robustness of the models. The authors do not explicitly indicate if their technique measures epistemic or aleatoric uncertainty (or both).

Ensemble Learning

Ensemble learning consists of combining multiple classifiers to create a stronger overall classifier (Opitz and Maclin 1999). Ensemble learning comes in three forms: bagging, boosting, and stacking. The reader is referred to (Opitz and Maclin 1999) for information about bagging and boosting. The technique we employ is stacking. Stacking is a technique that combine classifiers by using the outputs of the individual models of the ensemble as inputs to a final machine learning model (Opitz and Maclin 1999; Wolpert 1992). This final machine learning model learns how to process the predictions of the models in the ensemble. This allows the ensemble to learn conditions under which different base classifiers should have more influence.

The idea of creating an ensemble of neural networks is not new (Hansen and Salamon 1990). Historically, research focused on generating predictive networks with good performance that make different mistakes (Opitz and Shavlik 1996b; 1996a; Zhou and others 2001). Ensembles of neural networks have also been used for malware detection. Yan et al. (Yan, Qi, and Rao 2018) utilized an ensemble of deep learning techniques to classify software as benign or malware. This technique used a Long Short-Term Memory Network (LSTM) (Gers and others 1999) and a Convolutional Neural Network (CNN) (Krizhevsky and others 2012) with a Logistic Regression model (Kleinbaum and others 2002) learning the final prediction from the output of the deep learning technique.

Logistic Regression

Logistic regression is a classical machine learning technique that utilizes linear models for classification purposes (McCullagh and Nelder 1989). A general linear regression model can predict any positive or negative real value that could far exceed the bounds of the classification task. This, along with the discrete nature of classification tasks, make linear regression inadequate for classification.

To remedy this, the logistic regression model uses the sigmoid function to keep the output of the model in $[0, 1]$. This allows predictions to be interpreted as a probability. For multi-class classification, a one-vs-all approach can be applied to ascertain probabilities for each class.

Thus, for input vector $\mathbf{x}$ with $x_1, x_2, ..., x_n \in \mathbf{x}$, the learned model takes the form

$$f(\mathbf{x}) = \frac{1}{1 + e^{\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n}},$$

learning constant coefficients $\alpha_1, \alpha_2, ..., \alpha_n$.

Multimodal Learning

In the context of machine learning, a modality is collection of data that represents a distinct way of expressing a phenomenon (Ramachandram and Taylor 2017). For example, a modality can be defined by the representation of the data (e.g., text, audio, and video) (Liu and others 2018). In the context of our multimodal malware analysis, a modality is defined by the type of data collected, such as API calls or strings present in the Android APK file (Kim and others 2018). It is assumed that features in the same modality are related in some way. For example, we consider all the strings in the source code as its own modality; however, we do not consider text in the application's manifest XML file as stings. Instead, these strings are their own modality, since the text originated elsewhere.

In traditional multimodal learning, the individual modalities consist of data that are describing the same phenomenon but require completely different processing techniques (e.g., image and text data) (Almaadeed and others 2015; Tzirakis and others 2017; Poria and others 2016). In our case, a modality is a collection of data describing one aspect of the malware analysis problem. Utilizing modalities allows for the incorporation of domain knowledge, with each modality given its own hierarchical representation in a neural network (Kim and others 2018). This can allow more features to be learned from each modality and provided to a deep neural network, thus increasing the representation capacity of the network(Liu and others 2018). Second, this can increase the efficiency of the network. Each modality can be trained individually before being combined for classification (Kim and others 2018). It may be the case that using a fully connected neural network is not possible given the size of the feature space and the physical limitations of computing hardware. However, using a multimodal approach can reduce the toll on physical hardware while allowing use of all features.

Uncertainty Quantification in Deep Multimodal Neural Networks

Deep Multimodal Neural Network

The multimodal neural network (MMNN) implemented is inspired by (Kim and others 2018), who had success with this architecture. The MMNN has two primary stages: the feature extraction stage and the classification stage. The feature extraction stage consists of a separate neural network

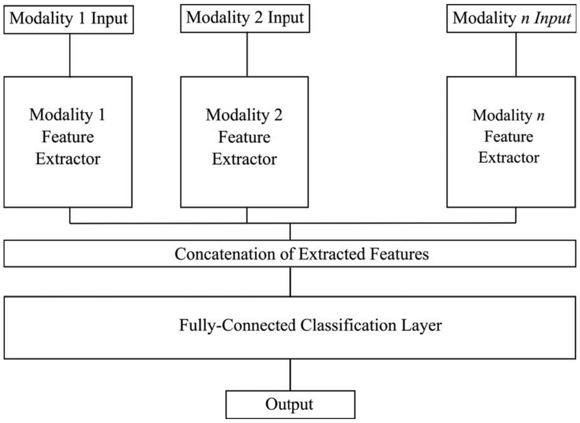
Modality 1 Input
Modality 2 Input
Modality *n Input*
Modality 1 Feature Extractor
Modality 2 Feature Extractor
Modality *n* Feature Extractor
Concatenation of Extracted Features
Fully-Connected Classification Layer
Output

Figure 1: Diagram of Multimodal Architecture

Stage	Layer	Size	Activation
Feature Extraction	Input Layer	N/A	N/A
	Hidden Layer	5000	ReLU
	Hidden Layer	2500	ReLU
	Hidden Layer	1000	ReLU
Classification	Hidden Layer	500	ReLU
	Hidden Layer	100	ReLU
	Hidden Layer	10	ReLU
	Output Layer	1	Sigmoid

Table 1: Architecture used to for Feature Extraction and Classification. This architecture is also used to pre-train the individual feature extractors

for each modality. The outputs of the last layer of each of the feature extractors are then combined to become the first layer of the multimodal classification stage. Although there are multiple encoding strategies for multimodal neural networks (e.g. (Chao and others 2015; Wang and others 2017)), we chose to follow the successful work of Kim et al. (Kim and others 2018) for this domain. A comparison of various encoding methods is beyond the scope of this work. The number of layers that process the feature extraction before concatenation is arbitrary; however, this architecture has been utilized previously in the literature to great success (Kim and others 2018). The classification layer then processes these extracted features into a final classification. Figure 1 and Table 1 give the architecture and parameters settings for the architecture.

Due to the size of the entire neural network, it is difficult to train the entire feature extraction stage and the classification stage concurrently. Thus, as a first step, for each modality, we train a unimodal neural network to pre-train the feature extractor stage. The architecture of the unimodal models trained are given in Table 1. The feature extraction stages of the unimodal models are combined to form the MMNN. Then, only the classification stage of the MMNN is trained.

Extracting Uncertainty Information

To extract the uncertainty from the multimodal model, we place persistent dropout layers before every hidden layer of the feature extraction stage. Then, once the feature extractors for the individual modalities have been combined, persistent dropout layers are placed before every hidden layer of the classification stage. We found that a dropout probability to 0.5 yielded the best accuracy. This means that each neuron in a dropout layer will have a probability of $p = 0.5$ of being set to 0 (Srivastava and others 2014). During the testing phase of the neural network, each datum was sampled through the network $T = 100$ times.

Dropout-Based Uncertainty Quantification in Deep Ensembles

Deep Ensemble

Our deep ensemble, referred to as "the ensemble" in the upcoming sections, combines using dropout to measure epistemic uncertainty (Gal and Ghahramani 2016) with the general framework of (Lakshminarayanan and others 2017). First, for each of the modalities, we train individual deep neural networks using the architecture given in Table 1. Persistent dropout is placed before every hidden layer with the exception of before the first hidden layer.

To combine the models together into an ensemble, let $\mathbf{p}$ be a vector of predicted probabilities from each of the unimodal models. Thus, for each $p_i \in \mathbf{p}$, $p_i \in [0, 1]$. Then, $p_j \in \mathbf{p}$ corresponds to the probability that the sampled datum has probability p_j of being an element of the positive class when sampled from the unimodal neural network responsible for modality j. The vector $\mathbf{p}$ is then provided as input to a logistic regression stacker to provide a final prediction. The logistic regression stacker is trained with the predictions of the unimodal models on training data and the labels of those data points.

Extracting Uncertainty Information

When an input x is provided to the model for prediction, each individual model of the ensemble is sampled $T = 100$ times. An ensemble prediction is generated for each of these $T = 100$ executions. From these ensemble predictions, a probability distribution is created from the predicted probabilities. We take the standard deviation of this probability distribution to ascertain an uncertainty value. Instead of predicting a variance from each model, which is consistent with aleatoric uncertainty (Kendall and Gal 2017) (Lakshminarayanan and others 2017), we used dropout to measure epistemic uncertainty. We chose to ascertain our uncertainty value by using the standard deviation of the ensemble predictions to correspond to measuring epistemic uncertainty from dropout.

Experimental Methodology

Dataset

For our experiments, we have a binary classification problem that can be formally described as follows: let our dataset be denoted as $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^M\}$, where M is the number of modalities in our dataset. Further, for each modality $m \in \{1, 2, ..., M\}$, it is the case that $\mathcal{D}^m = \{(\mathbf{x_i}, y_i)\}$ for $i \in \{1, 2, ..., N\}$, where N is the number of instances

Modality	**Number of Features**
Function Opcodes	2396
Manifest	140
Method API	974
Method Opcodess	4,968
String	39,512
Total	**47,990**

Table 2: Breakdown of features per modality

Class	**Number**	**Percent**
Malicious	1257	60.93%
Benign	806	39.07%

Table 3: Breakdown of class labels of this dataset

in $\mathcal{D}^m$. Further, the number of instances is the same for each modality. Since this is a binary classification problem, $y_i \in \{0, 1\}$. For modalities $\mathcal{D}^i$ and $\mathcal{D}^j$ where $i \neq j$, it is assumed that the feature sets are mutually exclusive. Further, for each modality $i \in \{1, 2, ..., M\}$, the dataset $\mathcal{D}^i$ can be split into $\mathcal{D}^i_{train}$ and $\mathcal{D}^i_{test}$ where $\mathcal{D}^i_{train} \cup \mathcal{D}^i_{test} = \mathcal{D}^i$ and $\mathcal{D}^i_{train} \cap \mathcal{D}^i_{test} = \emptyset$. Further, for $\mathcal{D}^i$ and $\mathcal{D}^j$ where $i \neq j$, it is the case that $\mathcal{D}^i_{train}$ and $\mathcal{D}^j_{train}$ contain the same instances. All modalities are trained and tested on the same instances.

The dataset that we utilize in our experiments contains $M = 5$ modalities detailing various features extracted from Android APKs (Kim and others 2018). The dataset totals 2,063 instances across a total of 47,990 features. We ran multiple experiments with this data and found that the Function Opcodes, String, and Manifest modalities outperformed all 5 modalties together. This led to a total of 42,048 features across the three chosen modalities. Feature breakdown by modality is given in Table 2. The class distribution is given in Table 3. Information regarding the modalities can be found in (Kim and others 2018).

Multimodal Neural Network

For performance evaluation, we trained each unimodal model and combined classification layers for 400 epochs, with early stopping enabled when a local maximum of the binary accuracy on the validation data was achieved after 50 epochs pass with no improvements. We use binary cross-entropy as the loss function and optimize using ADAM (Kingma and Ba 2014). We evaluated metrics for three executions of 10-fold stratified cross-validation. Statistical significance was calculated using values from each fold of cross-validation.

Ensemble Setup

The ensemble chosen was a stacking predictor created from the unimodal models used to pre-train the feature extraction stage of the multimodal neural network and a logistic regression model. We train and test the unimodal models with the same data as the multimodal neural network, and as such, when evaluated, each unimodal model is sampled $T = 100$ times with input data. All components of the ensemble are trained with the same hyperparameters as those for multimodal neural network. The logistic regression model is from the Scikit-learn library (Pedregosa and others 2011) and is trained using grid search to optimize the parameters.

Evaluation of Uncertainty Quantification

To evaluate the uncertainty quantification of the models, we use several metrics. First, we directly compare the average uncertainty values between models of individual folds. Each fold is equally split, and each model evaluated is trained and tested on the same instances. Thus, it is possible to directly compare the impact of each model on uncertainty.

The second evaluation technique of the uncertainty quantification is to compare how well the models extrapolate from training data to testing data. Average training uncertainty is calculated by averaging the uncertainty ascertained from the training data across all folds. Average testing uncertainty is calculated by averaging the uncertainty ascertained from the testing data across all folds. A measure of epistemic uncertainty should result in a larger testing uncertainty than training uncertainty (Gal and Ghahramani 2016). This is because epistemic uncertainty should increase when the model encounters data not seen in training. The model should be fairly confident in its predictions for the training data, since the model repeatedly processes the training data during training; however, in a situation when the model encounters unseen data, the epistemic uncertainty should be higher, since the model is relying on its own ability to inference rather than memory about seen data.

Experimental Environment

The models were created using the Keras deep learning library with Tensorflow backend (Chollet and others 2015; Abadi and others 2015). The models were trained and evaluated on a high performance computing cluster with an Intel Xeon E5-2680v4 CPU and an Nvidia Tesla K80 with 12GB graphics memory.

Results

Model Performance

In Table 4, it is evident that the ensemble is the most accurate of all the models tested. We use a paired T-test across each fold to verify statistical significance of the claim that the ensemble is the most accurate technique to at least 95% confidence. P-values are given in Table 5.

Model	Accuracy	F1 Score	AUC
Function Opcodes	0.6078	0.7555	0.4996
Manifest	0.9711	0.9766	0.9662
String	0.9439	0.9576	0.9365
MMNN	0.9471	0.9579	0.9855
Ensemble	**0.9814**	**0.9849**	**0.9910**

Table 4: Accuracy, F1-Score, and Area Under the Receiver Operating Curve of each model. Best in bold.

Model	P-Value
Function Opcodes	1.72E-45
Manifest	1.68E-08
String	4.45E-03
MMNN	2.28E-02

Table 5: P-values resulting from comparing the accuracy of the ensemble to the accuracy of the either a single modality or the mutli-modal neural network $n = 30$

Reducing Uncertainty

Table 6 contains the average uncertainty of the multimodal neural network and the ensemble model. Numerically, uncertainty is lowest in the ensemble and the highest in MMNN. A paired T-test on the uncertainty of each fold indicates that the ensemble reduces uncertainty from the String modality and the multimodal neural network with a confidence of 99%. Exact p-values are shown in Table 7.

Model	Average Uncertainty
Function Opcodes	0.0190799
Manifest	0.018638
String	0.0551729
MMNN	0.0892703
Ensemble	**0.0182969**

Table 6: Average Uncertainty of each model. Best in bold.

Model	P-Value
Function Opcodes	0.3512
Manifest	0.3926
String	*4.33E-06*
MMNN	*8.88E-09*

Table 7: P-values resulting from comparing the uncertainty of the ensemble to the accuracy of the either a single modality or the mutli-modal neural network

Extrapolation

Finally, we measured the potency of the uncertainty metrics between the multimodal and ensemble models by evaluating how much epistemic uncertainty varies from training uncertainty to testing uncertainty. Ideally, epistemic uncertainty in a model should be greater on testing data than on training data. This is an indicator that the uncertainty metric can successfully detect that the model has not been trained on given input.

To evaluate this, we took the average per-fold difference of training uncertainty and testing uncertainty for the multimodal network and the ensemble. These values are given in Table 8. It is evident that the ensemble model has a greatest uncertainty difference of all the models. Using the paired T-test on the individual folds as before, these differences are statistically significant with a confidence of at least 95%. Thus, our technique for measuring epistemic uncertainty appears to be more effective in ensembles of modalities rather than in MMNNs. Unfortunately, the Function Opcode modality does not experience the desired difference between test and train uncertainty. Undoubtedly, this results from the model's inability to infer from the function opcode data, as shown in Table 8.

Model	Extrapolation Value
Function Opcodes	-0.0008459
Manifest	0.0052349
String	0.0018848
MMNN	0.0033502
Ensemble	**0.006425651**

Table 8: Difference Between Average Test Uncertainty and Average Training Uncertainty for each model. Higher is better. Best in bold.

Model	P-Value
Function Opcodes	1.02E-08
Manifest	0.0475
String	3.21E-03
MMNN	0.0205

Table 9: P-values resulting from comparing the extrapolation values of the ensemble to the other models using a paired T-test. $n = 30$

Discussion

As evidenced in Table 4 and Table 5, our ensemble of modalities outperforms both the unimodal and hierarchical model (MMNN) with statistical significance. It is important to note that utilizing persistent dropout may reduce the accuracy of a model, since important neurons may be removed. Despite the usage of persistent dropout, however, our ensemble was able to achieve very high accuracy on this dataset.

The ensemble also had the lowest average uncertainty of the methods we examined; however, this result was statistically significant only for the String modality and the hierarchical multimodal network. This is likely because of the relatively large difference in uncertainty values. It is interesting to note that the Function Opcodes modality had relatively low uncertainty yet much lower accuracy. It is important to note that the accuracy of the Function Opcodes modality aligns with the percent of malicious instances. Thus, the low uncertainty results from the small amount of variation in predictions for instances.

Finally, the ensemble technique had a much higher difference between average testing uncertainty and average training uncertainty. Higher values of epistemic uncertainty typically imply that the model has not encountered some instance and might need human assistance in determining the final classification. It can be important that a model recognizes these instances effectively. As shown in Table 6 and Table 7, our ensemble technique had a higher extrapolation value than the other models. This means that the uncertainty value is more likely to detect unseen examples.

Conclusion and Future Work

In this work, we presented a epistemic uncertainty measure for ensembles of deep neural networks and hierarchical multimodal networks. Although the ensemble model does not statistically significantly reduce uncertainty over that in some of the individual modalities, this technique is able to improve accuracy over the both individual models from which it comprised and a hierarchical multimodal technique. Further, we show that this ensemble-based technique more clearly detects unseen examples compared to the other techniques.

In the future, we would like to apply this technique to other domains such as medical informatics and to additional datasets. Further, we would like to compare our uncertainty quantification approach in deep ensembles to other ensemble-based approaches. This work focused on an ensemble compromised of neural networks with deep architectures. Thus, studying the applicability of this work to shallow architectures is a possible future direction. Finally, we would like to combine both epistemic and aleatoric uncertainties in an ensemble-based technique.

References

Abadi, M., et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Almaadeed, N., et al. 2015. Speaker identification using multimodal neural networks and wavelet analysis. *IET Biometrics* 4(1):18–28.

Brandom, R. 2019. There are now 2.5 billion active android devices. *The Verge*.

Brown, K., and Talbert, D. 2019. Estimating uncertainty in deep image classification. In *Proceedings of the American Medical Informatics Association Annual Symposium*.

Chao, L., et al. 2015. Long short term memory recurrent neural network based multimodal dimensional emotion recognition. In *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, 65–72.

Chollet, F., et al. 2015. Keras.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Gers, F. A., et al. 1999. Learning to forget: Continual prediction with lstm. *Ninth International Conference on Artificial Neural Networks ICANN 99*.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (10):993–1001.

Kendall, A., and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Kim, T., et al. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14(3):773–788.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization.

Kleinbaum, D. G., et al. 2002. *Logistic regression*. Springer.

Krizhevsky, A., et al. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lakshminarayanan, B., et al. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Leibig, C., et al. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports* 7(1):17816.

Liu, K., et al. 2018. Learn to combine modalities in multimodal deep learning. *arXiv preprint arXiv:1805.11730*.

McCullagh, P., and Nelder, J. A. 1989. Generalized linear models.

Michelmore, R., et al. 2018. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*.

Opitz, D., and Maclin, R. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* 11:169–198.

Opitz, D. W., and Shavlik, J. W. 1996a. Actively searching for an effective neural network ensemble. *Connection Science* 8(3-4):337–354.

Opitz, D. W., and Shavlik, J. W. 1996b. Generating accurate and diverse members of a neural-network ensemble. In *Advances in neural information processing systems*, 535–541.

Pedregosa, F., et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Poria, S., et al. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *2016 IEEE 16th international conference on data mining (ICDM)*, 439–448. IEEE.

Ramachandram, D., and Taylor, G. W. 2017. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine* 34(6):96–108.

Srivastava, N., et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Tzirakis, P., et al. 2017. End-to-end multimodal emotion recognition using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing* 11(8):1301–1309.

Wang, X., et al. 2017. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5239–5247.

Wolpert, D. H. 1992. Stacked generalization. *Neural networks* 5(2):241–259.

Yan, J.; Qi, Y.; and Rao, Q. 2018. Detecting malware with an ensemble method based on deep neural network. *Security and Communication Networks* 2018.

Zhou, Z., et al. 2001. Genetic algorithm based selective neural network ensemble. In *IJCAI-01: proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, Washington*.