Hybrid End-to-End Convolutional K-Nearest Neighbors Classifier

Abstract

This paper introduces a novel end-to-end machine learning model that replaces the fully-connected layers of a Convolutional Neural Network (CNN) with a differentiable K-Nearest Neighbor (KNN) classifier. Due to the large number of weights that need to be learned in a fully connected layer, fully connected neural networks are computationally expensive to train, require a large amount of data to train accurately, and are difficult to interpret. By replacing the fully connected layers of a traditional CNN with a differentiable KNN, our hybrid Convolutional K-Nearest Neighbors (C-KNN) classifier overcomes the limitations of fully connected layers without a reduction in classification accuracy.

Introduction

Recent literature has shown that CNNs can produce remarkable, state-of-the-art results across a multitude of domains and applications; however, like other artificial neural networks (ANN), they are computationally expensive, data hungry, and often considered black box models due to a lack of interpretability (Aloysius and Geetha 2017; Rawat and Wang 2017; Anwar et al. 2018). The success of CNNs, and often ANNs in general, has fallen short in domains and applications where collecting large datasets is infeasible or where trust and interpretability in the model is of utmost importance such as the medical field. In the medical field, data can be hard to come by due to confidentiality agreements, lack of expert annotations, and inconsistent data points and trust in the model needs to be the highest priority as people's lives are heavily affected by the diagnoses and treatments they receive (Asan et al. 2020; Ravì et al. 2017; Miotto et al. 2017). It has been shown that human trust in a model is often dependent on its explainability or interpretability as well as its usability and reliability (Glikson and Woolley 2020; Siau and Wang 2018). To build this trust, we propose replacing the cause of these issues, the fully connected layers, with the KNN algorithm that is often much simpler to interpret and requires less computational power for its use while still providing comparable accuracies (Taunk et al. 2019; Tayeb et al. 2017).

The traditional CNN architecture is composed of convolutional layers used for feature extraction that are flattened and fed into fully-connected layers used for the classification task. This paper introduces a hybrid end-to-end C-KNN model that replaces the fully-connected layers with a differentiable K-Nearest Neighbors classifier. Thus, the C-KNN model doesn't suffer from the black box and computationally expensive nature of fully-connected layers while proving to be more robust to small training sets, requiring less architecture and hyper-parameter optimization, and maintaining comparable classification performance to the traditional CNN models.

Related Work

Feature Learning

Hybrid models using traditional CNNs to learn and extract features has been proposed and implemented, but never in a true end-to-end fashion that replaces the fully-connected layers. (Niu and Suen 2012; Ahlawat and Choudhary 2020) both propose a hybrid CNN-SVM classifier where a CNN is initially trained with its fully-connected layers and then the intermediary features are fed into an SVM for classification. These hybrid CNN-SVM classifiers inherit the limitations of fully-connected layers and the weights learned by the convolutional layers may not be optimal for use in an SVM. Similar work using features trained for KNNs has been done as well, with the same issues as the CNN-SVM models (Ren et al. 2014). On the contrary, our C-KNN model completely replaces the fully-connected layers and learns weights directly for the differentiable KNN.

Differentiable KNN

Different forms of a differentiable KNN have been proposed and implemented, but never used to replace the fully-connected layers in a neural network. (Plötz and Roth 2018) proposes a KNN layer with a differentiable selection rule that can be inserted into existing ANNs. This KNN layer is designed to work alongside fully-connected layers to learn non-local features that convolutional layers cannot. (Ren et al. 2014) instead proposes a differentiable loss function tailored to the KNN task which we later use and adapt in our own architecture.

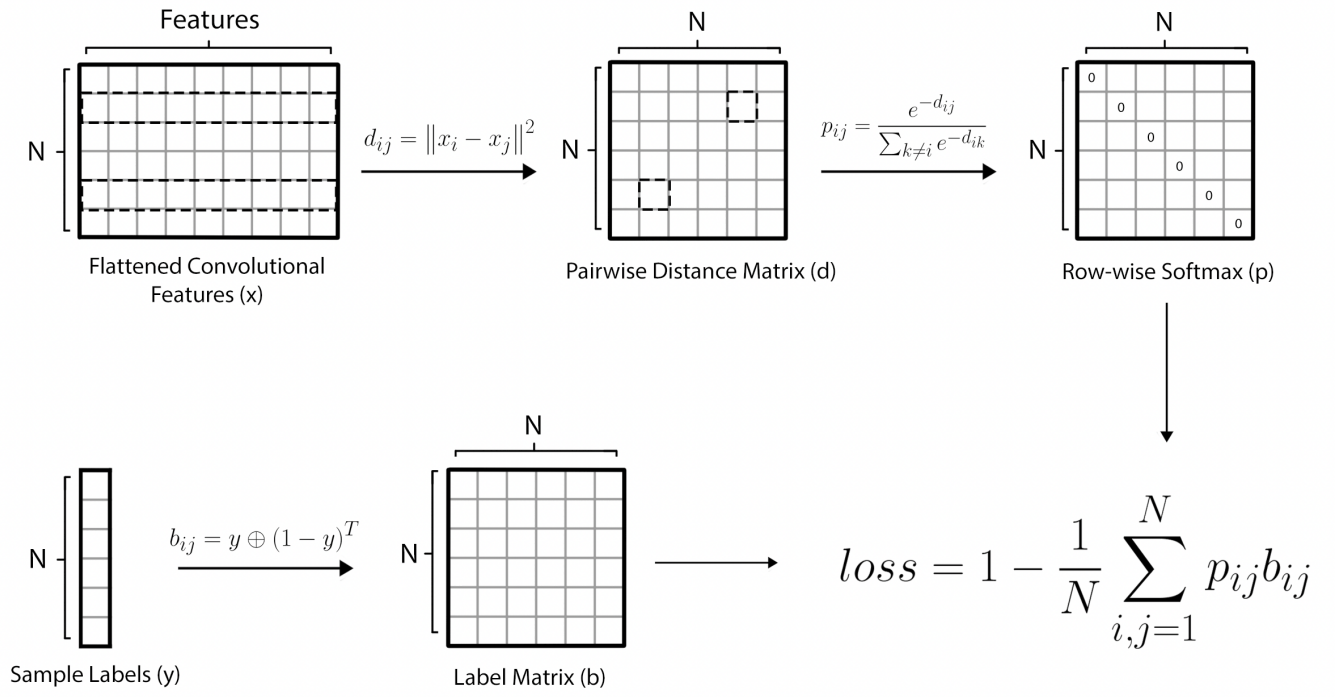

Figure 1: Differentiable C-KNN Forward Propagation

Metric Learning

Metric learning is a method used for KNN algorithms that finds a distance metric that is most suitable for the task given, eliminating the need for a manual definition of a distance metric. As KNNs can vary in performance depending on the distance metric used (Prasath et al. 2017), metric learning can be a powerful tool to use. Our method performs metric learning closest to Neighbourhood Component Analysis (NCA) as the convolutional layers of our architecture would be equivalent to the learned transformation matrix from NCA (Goldberger et al. 2004). NCA can be seen as a linear stochastic variant of KNN and extensions have been made to extend NCA for nonlinear applications (Salakhutdinov and Hinton 2007) as well as for values of k other than k=1 (Tarlow et al. 2013).

Methodology

Overview

Our hybrid C-KNN model replaces the fully-connected layer(s) of the standard CNN with a differentiable KNN layer in an attempt to remove the limitations that come with fully-connected layers. We implement both our hybrid C-KNN and a CNN using Python and the NumPy and TensorFlow libraries. Both models utilize convolutional and pooling layers implemented with TensorFlow's raw_ops module which contains functions to compute both the forward and backward propagation for each operation. The implementation of these layers is identical in both models to ensure a fair comparison. For the traditional CNN, the output of the convolutional layers is flattened and fed into fully-connected layers implemented with standard NumPy operations. For our hybrid C-KNN, the output of the convolutional layers is flattened and fed into a differentiable KNN layer implemented with standard NumPy and TensorFlow operations. Both models use stochastic gradient descent as an optimization algorithm and have weights initialized from a random normal distribution. The traditional CNN uses binary cross-entropy as a loss function, while the hybrid C-KNN uses an NCA inspired loss function (Ren et al. 2014) that motivates the convolutional layers to learn features better suited for a k=1 KNN classification.

Differentiable KNN

Figure 1 shows the overall process of forward propogation through the C-KNN Our method utilizes NCA (Ren et al. 2014) as a loss function to motivate the convolutional layers to learn weights optimized for KNN. Given N training examples and their labels $\{(x_i, y_i)|i = 1, ..., N\}$, first calculate an NxN pairwise distance matrix using the distance function

$$d(x_i, x_j) = \|F(x_i) - F(x_j)\|^2 \tag{1}$$

where F(.) is the transformation done by the convolutional layers of the network. Using the distance function, we compute p_{ij}, the probability of an example x_i inheriting its class from another example x_j or the row-wise softmax values for each distance in a row. p_{ij} is defined as

$$p_{ij} = \frac{e^{-d(x_i, x_j)}}{\sum_{k \neq i} e^{-d(x_i, x_k)}} \quad , \quad p_{ii} = 0 \tag{2}$$

Method		CNN			C-KNN (Ours)					
Train Size	Batch Size	α	Ex. Time	Accuracy	α	Ex. Time	2115	1057	Batch Size	15
12665	1024	0.01	60	99.34	10	89	99.89	99.93	99.93	99.33
6332	1024	0.01	30	99.04	10	44	99.90	99.91	99.91	99.31
160	1024	0.01	0.88	93.31	10	0.77	99.79	99.89	99.80	99.07
12665	512	0.01	62	99.61	1	77	99.90	99.91	99.88	99.36
6332	512	0.01	31	99.33	1	38	99.89	99.90	99.86	99.39
160	512	0.01	0.80	92.13	1	0.78	99.90	99.92	99.88	99.32
12665	32	0.002	70	99.64	0.01	73	99.89	99.95	99.63	99.30
6332	32	0.002	36	99.49	0.01	38	99.86	99.95	99.60	99.21
160	32	0.002	0.91	97.50	0.01	0.91	99.89	99.91	99.65	99.39

Table 1: Accuracy comparison of our method and a traditional CNN on the Handwritten Digits dataset.

The loss function can now be expressed using the probability of example x_i being correctly classified in class c as

$$Loss_{NCA} = 1 - \frac{1}{N}\sum_{i=1}^{N}\sum_{y_j=c} p_{ij} = 1 - \frac{1}{N}\sum_{i,j=1}^{N} p_{ij}b_{ij} \quad (3)$$

where $b_{ij} = 1$ if $y_i = y_j$ and $b_{ij} = 0$ if $y_i \neq y_j$. This NCA loss function in Eqn. 3 is continuous and differentiable, allowing us to use it for our end-to-end C-KNN.

Since the NCA loss function aims to maximize the probability that example (x_i, y_i) is correctly classified into class y_i, our final classification for an example is determined by selecting the single neighbor with the highest probability p_{ij} and inheriting its class label.

Backpropagation. Now that our forward propagation is differentiable, we can utilize TensorFlow's GradientTape module for automatic computation of gradients. GradientTape records any TensorFlow operations performed within it and uses the stored information to calculate the gradients we need at each step in order to update the weights during backpropagation.

Experiments

Datasets and Model Architectures

Each model architecture consists of a number of convolutional and max pooling layers followed by our custom KNN layer or by two fully connected layers, the first using a ReLU activation and the second using a sigmoid activation for final classification.

Handwritten Digits MNIST is a dataset with 10 classes of 28x28 greyscale images of the handwritten digits 0-9. For the purposes of binary classification, only the classes labeled as 0 and 1 are used for testing and training, corresponding to zeros and ones respectively. The training set of only 0's and 1's contains 12665 images total with 5923 being 0's and the remaining being 1's and the test set contains 2115 images with 980 being 0's and the remaining being 1's. The model architecture consists of one 3x3 convolutional layer with 32 filters followed by one 3x3 max pooling layer. The traditional CNN model used for comparison is flattened into a 32 neuron fully-connected layer followed by a single neuron fully-connected layer, while the hybrid C-KNN is just fed into the differentiable K-Nearest Neighbors classifier.

Fashion MNIST is another MNIST dataset which contains 10 classes of 28x28 greyscale images of clothing items. To simplify the classes into a binary problem, only the classes labeled as 0 and 1 are used for testing and training, corresponding to t-shirts and trousers respectively. The training and test sets for this dataset are split evenly for our chosen classes and contain a total of 12000 and 2000 images respectively. The model architecture consists of two 3x3 convolutional layers, the first with 32 filters and the second with 64, each followed by a 2x2 max pooling layer. The traditional CNN model used for comparison is flattened into a 128 neuron fully-connected layer followed by a single neuron fully-connected layer, while the hybrid C-KNN is just fed into the differentiable K-Nearest Neighbors classifier.

Horses or Humans is a binary classification dataset with 300x300 generated colored images of horses and humans. There are a total of 1027 images in the training set with 500 examples being the horse class and 527 being the human class. The test set contains 256 images with 128 images of each class. Since this is a more complex problem with larger images, our architecture uses more convolutions than for the MNIST datasets. We use five 3x3 convolutional layers, with 16 filters for the first layer, 32 filters for the second layer, and 64 for the rest, each followed by a 2x2 max pooling layer. The traditional CNN model used for comparison is flattened into a 512 neuron fully-connected layer followed by a single neuron fully-connected layer, while the hybrid C-KNN is just fed into the differentiable K-Nearest Neighbors classifier.

Evaluation

We evaluate performance on the datasets using accuracy and execution time required to train for each model architecture. Since the C-KNN's classification is affected by the number of neighbors each sample has to choose from, we run the test set for C-KNN using 4 different batch sizes and report the average accuracy for all the batches. The traditional CNN is not affected by the number of samples in a batch and reports the same accuracy score across different batches so we only report a single accuracy score. All accuracies and execution times are averaged over the training and testing of 10 different models using the same model architecture. We train each model for 10 epochs and choose the largest learning rate, α,

Method		CNN			C-KNN (Ours)					
Train Size	Batch Size	α	Ex. Time	Accuracy	α	Ex. Time	2000	1000	Batch Size	15
12000	1024	0.0001	82	94.69	10	110	96.64	95.89	96.16	87.41
6000	1024	0.0001	37	92.90	10	56	96.96	96.17	96.50	87.84
160	1024	0.0001	1.2	84.86	10	1.29	96.50	96.03	92.87	87.34
12000	512	0.0001	87	96.22	1	102	97.30	96.78	96.72	88.50
6000	512	0.0001	40	95.26	1	51	97.22	96.69	96.81	88.89
160	512	0.0001	1.24	83.33	1	1.3	96.64	96.02	92.70	87.32
12000	32	0.00001	110	96.86	0.01	111	95.91	95.40	88.22	86.27
6000	32	0.00001	54	95.51	0.01	51	96.29	95.75	88.51	86.73
160	32	0.00001	1.5	78.38	0.01	1.37	97.02	96.62	91.02	88.87

Table 2: Accuracy comparison of our method and a traditional CNN on the Fashion dataset.

Method		CNN			C-KNN (Ours)					
Train Size	Batch Size	α	Ex. Time	Accuracy	α	Ex. Time	2000	1000	Batch Size	15
1027	64	0.005	804	87.73	0.05	771	99.38	96.33	95.35	86.44
513	64	0.005	393	73.09	0.05	377	98.91	97.77	95.98	90.48
160	64	0.005	97	50.00	0.05	118	98.59	97.54	96.05	91.72
1027	32	0.0025	786	90.55	0.02	723	98.91	95.86	92.89	86.04
513	32	0.0025	390	75.86	0.02	357	99.34	97.07	94.26	88.52
160	32	0.0025	120	50.00	0.02	111	98.59	97.70	94.65	91.12
1027	16	0.00125	795	89.41	0.0025	735	99.49	97.97	88.24	88.04
513	16	0.00125	397	82.11	0.0025	352	99.18	98.09	90.27	90.36
160	16	0.00125	121	50.00	0.0025	115	98.67	97.81	91.29	91.44

Table 3: Accuracy comparison of our method and a traditional CNN on the Horses or Humans dataset.

that we can for each training batch size that still allows for training with the full training set. Keeping the learning rate consistent for batch sizes allows us to showcase the effect that the size of your dataset has on the two models.

Results

Evaluation on Handwritten Digits MNIST

As shown in Table 1, we compare our C-KNN method to our baseline traditional CNN with dense layers. Since the handwritten digits dataset is fairly simple, the differences between the two models are fairly small. However, our method still outperforms the CNN by 0.27% when comparing their highest accuracies overall. We can also observe that while the CNN's accuracy decreases as less data is provided for training, the C-KNN's accuracy does not change much at all, making it more robust for smaller training sets.

Evaluation on Fashion MNIST

In the fashion dataset we start to observe larger differences between our method and the CNN. As illustrated in table 2, our method outperforms the CNN by 0.44% when comparing their highest accuracies overall. The C-KNN also begins to show more differences between it's accuracies of different testing batch sizes, with smaller batches resulting in a lower accuracy. This is likely because of the 1-NN having more options to choose from when the batch size is larger. Along with batch sizes, the training set size has less of an effect on the model's performance for the C-KNN versus the CNN with the C-KNN differing by an average of 0.64% between the highest and lowest training set size and the CNN differing by an average of 13.73%.

Evaluation on Horses or Humans

The Horses or Humans dataset shows us the largest differences out of the three datasets we tested. As shown in Table 3, our method achieves a highest accuracy that is 8.94% higher than the CNN's highest accuracy overall. Similar to the fashion dataset, the C-KNN also does not require as much training data in order to achieve high accuracies while the CNN predicts at random without more hyperparameter tuning. Lastly, the accuracies for the C-KNN follow an interesting pattern in that the accuracies for larger batch sizes increase as more training occurs while the accuracies for smaller batch sizes decreases as more training occurs. Again, this indicates that the C-KNN works better as it has more neighbors to chose from for its prediction.

Conclusion

We have created a novel hybrid end-to-end architecture for image classification tasks. This architecture, called a C-KNN, eliminates the limitations of a traditional CNN by replacing the dense layers with a modified version of 1-nearest neighbor algorithm. Our method achieves significantly higher accuracy than a traditional CNN while requiring less computations and eliminating the "black box" effect from too many calculations. Removing these two limitations of the traditional CNN can help encourage trust in

these models in more tasks since many professionals are hesitant to use something that will take too long to realistically run or cannot explain why it did what it did.

Future Work

The work shown here is an initial proof-of-concept, offering significant advantages to current CNN's albeit on limited data and fairly simple tasks. Future work will focus primarily on showing how C-KNN's can be used in complex data analysis tasks and as building blocks for more complex learning systems. Further, there are many facets of this novel hybrid end-to-end C-KNN classifier that remain to be explored and improved. The squared euclidean distance is currently used as the distance metric of the differentiable KNN and that is fed into an NCA-based loss function. The impact of different combinations of distance metrics and loss functions remain to be explored, and may be able to motivate the learner in a more effective way. The optimization algorithm that currently drive learning in the C-KNN is NCA loss; however, different optimization algorithms have proven to be effective in traditional CNNs, so their impact on the C-KNN may also be promising.

References

Ahlawat, S.; and Choudhary, A. 2020. Hybrid CNN-SVM Classifier for Handwritten Digit Recognition. *Procedia Computer Science*, 167: 2554–2560. International Conference on Computational Intelligence and Data Science.

Aloysius, N.; and Geetha, M. 2017. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, 0588–0592.

Anwar, S. M.; Majid, M.; Qayyum, A.; Awais, M.; Alnowami, M.; and Khan, M. K. 2018. Medical image analysis using convolutional neural networks: a review. *Journal of medical systems*, 42(11): 1–13.

Asan, O.; Bayrak, A. E.; Choudhury, A.; et al. 2020. Artificial intelligence and human trust in healthcare: focus on clinicians. *Journal of medical Internet research*, 22(6): e15154.

Glikson, E.; and Woolley, A. W. 2020. Human Trust in Artificial Intelligence: Review of Empirical Research. *Academy of Management Annals*, 14(2): 627–660.

Goldberger, J.; Roweis, S.; Hinton, G.; and Salakhutdinov, R. 2004. Neighbourhood Components Analysis. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, 513–520. Cambridge, MA, USA: MIT Press.

Litjens, G.; Kooi, T.; Bejnordi, B. E.; Setio, A. A. A.; Ciompi, F.; Ghafoorian, M.; van der Laak, J. A.; van Ginneken, B.; and Sánchez, C. I. 2017. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42: 60–88.

Miotto, R.; Wang, F.; Wang, S.; Jiang, X.; and Dudley, J. T. 2017. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6): 1236–1246.

Niu, X.-X.; and Suen, C. Y. 2012. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4): 1318–1325.

Plötz, T.; and Roth, S. 2018. Neural Nearest Neighbors Networks. *CoRR*, abs/1810.12575.

Prasath, V. B. S.; Alfeilat, H. A. A.; Lasassmeh, O.; and Hassanat, A. B. A. 2017. Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier - A Review. *CoRR*, abs/1708.04321.

Ravì, D.; Wong, C.; Deligianni, F.; Berthelot, M.; Andreu-Perez, J.; Lo, B.; and Yang, G.-Z. 2017. Deep Learning for Health Informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1): 4–21.

Rawat, W.; and Wang, Z. 2017. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9): 2352–2449.

Ren, W.; Yu, Y.; Zhang, J.; and Huang, K. 2014. Learning Convolutional Nonlinear Features for K Nearest Neighbor Image Classification. In *2014 22nd International Conference on Pattern Recognition*, 4358–4363.

Salakhutdinov, R.; and Hinton, G. 2007. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In Meila, M.; and Shen, X., eds., *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, 412–419. San Juan, Puerto Rico: PMLR.

Siau, K.; and Wang, W. 2018. Building trust in artificial intelligence, machine learning, and robotics. *Cutter business technology journal*, 31(2): 47–53.

Tarlow, D.; Swersky, K.; Charlin, L.; Sutskever, I.; and Zemel, R. 2013. Stochastic k-Neighborhood Selection for Supervised and Unsupervised Learning. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 199–207. Atlanta, Georgia, USA: PMLR.

Taunk, K.; De, S.; Verma, S.; and Swetapadma, A. 2019. A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 1255–1260.

Tayeb, S.; Pirouz, M.; Sun, J.; Hall, K.; Chang, A.; Li, J.; Song, C.; Chauhan, A.; Ferra, M.; Sager, T.; Zhan, J.; and Latifi, S. 2017. Toward predicting medical conditions using k-nearest neighbors. In *2017 IEEE International Conference on Big Data (Big Data)*, 3897–3903.

Xiao, C.; Choi, E.; and Sun, J. 2018. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10): 1419–1428.