CS6140 Final Project Document
Shubham Bansal
Vansh Gupta

Topic - Street Image Cleanliness Classification

Abstract: This project is aimed towards building a robust model for automatically assessing the cleanliness of streets in images using convolutional neural networks (CNNs). The dataset comprises two distinct classes: clean and dirty streets. Various CNN architectures, including a simple custom CNN, ResNet, and Shufflenet, will be employed for image classification. The models will be trained on labeled data to learn discriminative features associated with clean and dirty street environments. The comparative analysis of different CNN architectures will shed light on their respective performances in addressing this specific image classification task. The outcomes of this project have the potential to contribute to advancements in computer vision applications for smart cities and sustainable urban development.

1. **Overview:**
   The goal of this project is to build machine learning models to classify images as clean or dirty and improve the overall infrastructure and sanitation status of a city by determining dirty streets proactively using technology and taking actions to clean them. This problem is interesting because it can significantly improve the infrastructure, and improve tourism, business and investment in any city. It can also improve the overall health and sanitation of the people and vital for developing regions of the world as ratio of unclean to clean streets is quite high in such regions. Ground breaking technologies like neural networks can really make the task efficient by eliminating the human factor in determining cleanliness of streets. The ultimate vision of such a project is to develop algorithms to identify dirty streets, then stream real-time dash cam footage from public service vehicles like ambulances, fire brigades, and police cars to capture images of these streets. Report the dirty street coordinates to authorities for prompt action.

   The approaches that we used to tackle the problem are discussed in the following section. We collected data from multiple sources like different Kaggle datasets, splitting youtube videos frame by frame etc. After data collection we performed data augmentation methods like rotation, flipping, cropping, adding Gaussian noise to prepare the dataset for training. We then planned to use three different models to perform the training on the training dataset. The first model is a simple CNN architecture which consists of three or four convolutional layer to see how good it can perform on a task like this. Secondly we used a prebuilt ResNet 18 model with 18 layers and lastly, we used a shufflenet model which is a lightweight CNN because the ultimate vision of this project is to run the models on small devices like dashcams or mobile phones. The rationale behind using ShuffleNet over ResNet50 is that we had limited dataset and to avoid overfitting, also because of limited compute resources we needed a lightweight architecture like shufflenet. Using ResNet18 gave us an accuracy of 95% and we skipped training a ResNet50 because it would have easily overfitted on 5000 images.

The Simple CNN that we built did not perform that great and saturated at an accuracy of 75% even after tweaking multiple parameters. We limited it to a 3 layer architecture to make it not too complex.

2. **Experiment setup**:
   The dataset consists of a total of 5000 street view images which are further split into a subsection of dirty and clean street images. In the notebook, using the directory structure, labels are provided to the image specifying whether it's clean or not.

   The implementation consists of different steps such as data augmentation, model definition, hyperparameter tuning and plotting the model performance.
   1. Data augmentation: It contains measures flipping images vertically, horizontal and rotating them to provide a varied dataset which will result in the model being able to adapt to alien data well.
   2. Model definition: We've implemented a custom CNN along with two pre-trained models, Resnet-18 and ShuffleNetV2. We run these models on a CPU chip.
   3. Hyperparameter tuning: We've tried and tested different parameters such as using different loss and optimization functions, tweaking the number of hidden layers in the custom Convolutional network and trying different activation functions.

   We've tried three different models for this project:-
   Custom CNN: We've built a three layer CNN with a pooling layer stacked after each convolutional layer. We've skipped out on using an activation function as the model seemed to perform better without. Lastly, we've used a couple of liner layers, sandwiched between a ReLU function.
   Pre-trained Resnet18 Model: We've used a predefined and trained Resnet18 model where we have changed the fully connected layer to result in an output of two features as we're performing binary classification.
   Pre-trained ShuffleNetV2 Model: We've used a predefined and trained ShuffleNet model where we have changed the fully connected layer to result in an output of two features as we're performing binary classification.

3. **Experiment results:**.
   The results tell us that, Resnet-18 performed the best with the highest accuracy achieved of 95 percent.
   Secondly, all the models seem to perform better after data augmentation rather than training models directly on the data which indicates that images collected are similar and fare well when randomness is added in.
   Models with complex architectures seem to perform poorly as it may overfit when dealing with sparse training data. There's a possibility of the model memorizing outliers within the training set, which results in overfitting.

   The parametric choices include number of layers, learning rate, number of epochs, loss function, and optimization function.

Number of layers: We tried combinations of 2, 3 and 4 layers for the Custom CNN, however, 2 layers seem to underfit while 4 layers seem to overfit.
Number of epochs: The pretrained models required about 10 epochs for optimal accuracy before they started overfitting. Custom CNN had to learn from scratch and continued improving model performance for about 20 epochs.
Learning rate: 0.001 for CNN and ShuffleNet and 0.0001 for Resnet18. The models performed differently based on the learning rate.
Loss function: Tried out logits loss and cross entropy loss functions which are suited for classification where cross entropy loss function fared better.
Optimization function: SGD and Adam Optimizer were adopted where faster convergence was provided by the Adam optimizer function.

4. **Discussion:**
   The Simple custom CNN that we built with 3 and 4 convolutional layers didn't seem to perform that well, possibly because the dataset was a little complicated for such a simple architecture. Resnet18 performed pretty well with an accuracy clocking at 95% and similarly Shufflenet clocked an accuracy of 94%. For this particular use case in real world, it would make a lot of sense to use a lightweight model like Shufflenet so that this can run on edge devices like dashcams in cars and provide real time updates on the cleanliness of streets in a city. We can make this more impactful by actually providing a framework to run this model on a small dashcam device and pull real time updates into a system which can then trigger an event to the authorities to take necessary action.
5. **Conclusion:**
   Using this project, cities can achieve a better infrastructure by providing a clean and healthy environment and develop an overall better economy and a better place to live for the people.
6. **References:**
   Dataset 1 - Link
   Dataset 2 - Link
   Research papers - Shufflenet, Resnet