CLASSIFICATION OF INDIAN ARCHITECTURAL DESIGNS

MACHINE LEARNING PROJECT REPORT

Submitted By

CHIRAN JEEVI (2019103013)

SWETA CHAKRAVARTHI (2019103070)

ADITYA RAMACHANDRAN (2019103502)

COMPUTER SCIENCE AND ENGINEERING COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

ABSTRACT

Each and every state in India has its own magnificent architecture and to be able to classify each style would be of great significance to architectural and historic studies. Convolutional Neural Networks (CNN) have a track record of exhibiting promising results for architecture type classification due to their feature extraction capability.

In this project the main focus initially lies in collecting the dataset. Data collection and initial pre-processing was done manually, amounting to roughly 1700 images. Images in this dataset consist of various Architectural Styles pertaining to 9 classes, namely Cave-Architecture, Chola, Hoysala, Kalinga, Kerala, MaruGurjara, Mughal, Rajput and Tibetan.

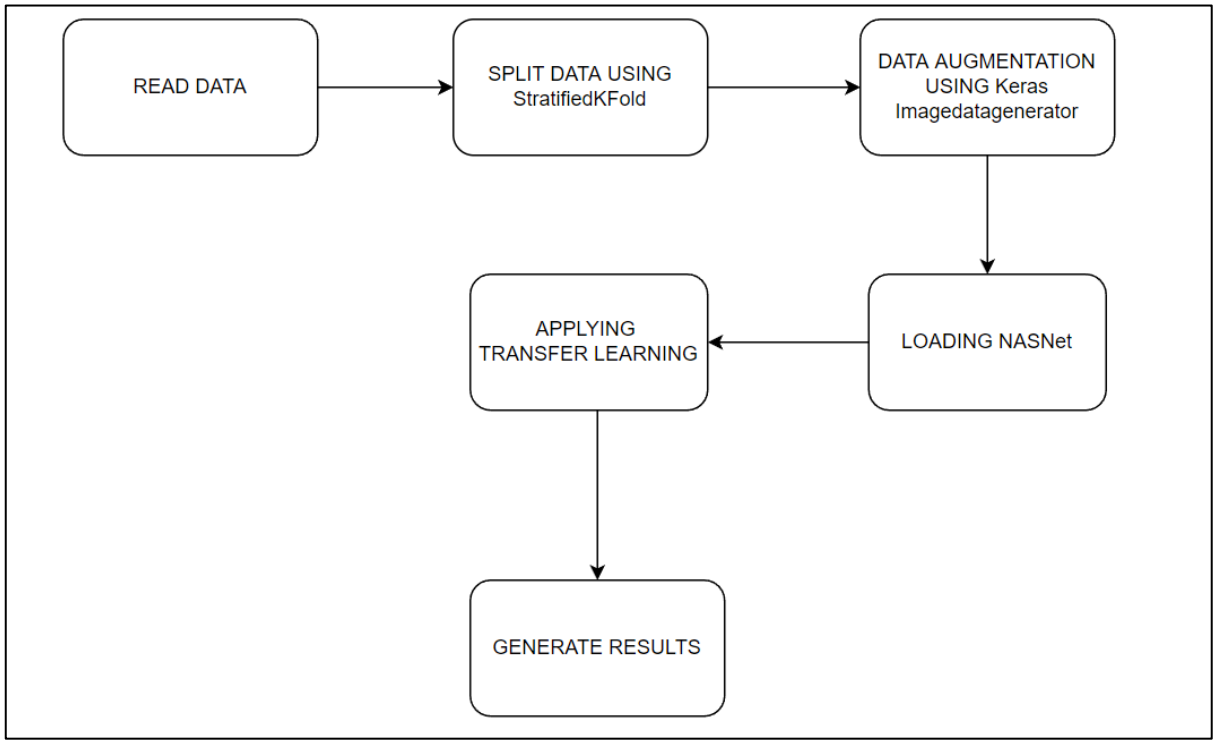
The next part of the project focuses on using this dataset and feeding it as input to various Machine Learning techniques and analysing which one is better at classifying the different architectural styles correctly. First off we work on using CNN with a combination of Support Vector Machine (SVM) algorithm for better classification of architectures. Then we check out Keras InceptionV3 architecture and analyse how it performs with the data.

Fast-AI is a deep learning library which has high level components that give very good results. This is due to its layered architecture which has common underlying patterns of various deep learning techniques in terms of decoupled abstractions. We try the Fastai ResNet and VGG models to analyse how it classifies the architectural images accurately.

METHODOLOGY AND BLOCK DIAGRAMS

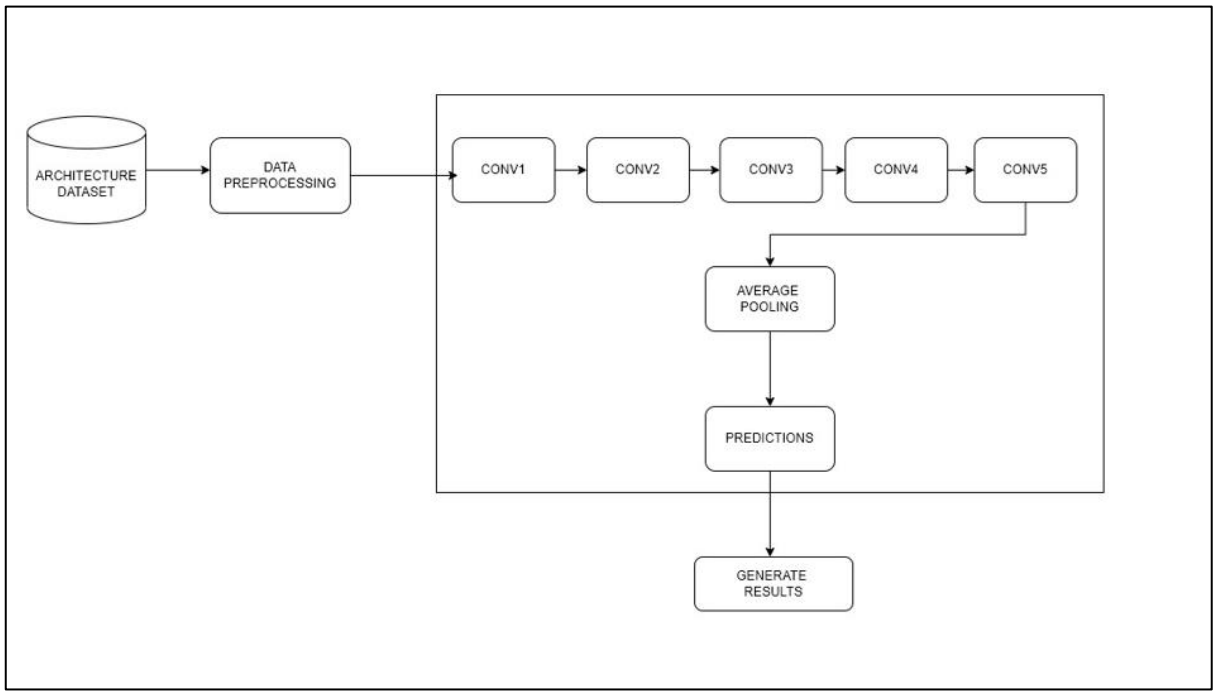
1. CNN-SVM:

- The main concept behind this approach is that deep learning is being used for feature extraction.Then, SVM model is used at the end for classification.
- Loading & reading data files (CSV)
- Then we divide the data into valid and training parts using StratifiedKFold as data is skewed.
- Since the dataset is quite large we use a datagenerator, Keras Imagedatagenerator, for data augmentation.
- Loading NASNet large model and setting all layers except last 35 as non-trainable so we can generalize our model on our data
- Then we perform transfer learning for better results



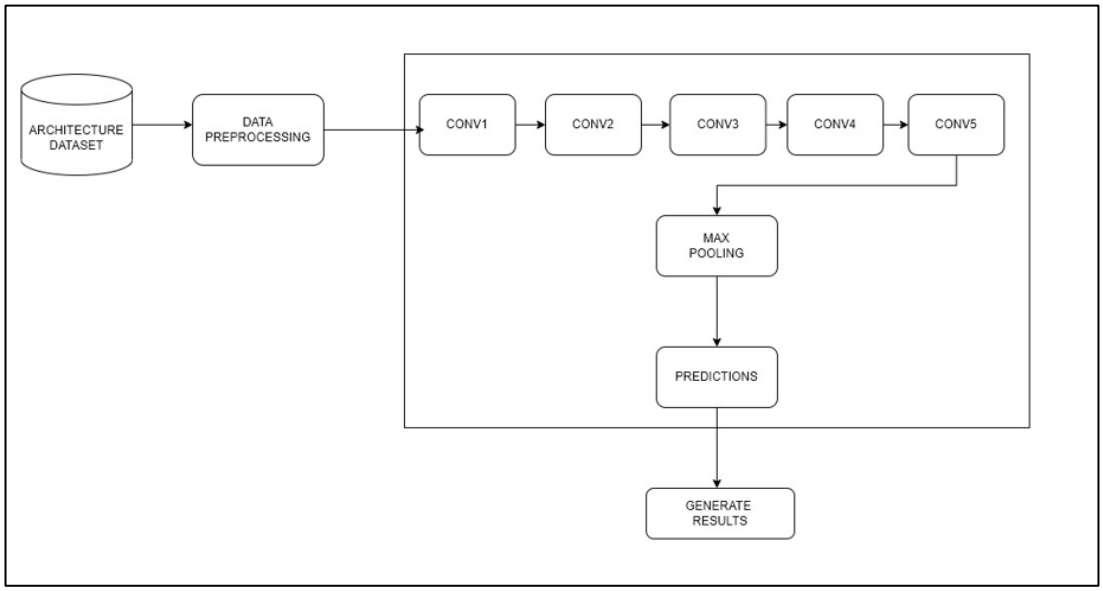
2. ResNet:

- First, we import the FastAI module and its APIs. These APIs help in building the architecture of the ResNet model.
- The pretrained FastAI-ResNet model is imported.
- Hyperparameters and other details are specified.
- The model is re-trained with the pre-processed dataset
- Accuracy and other metrics are displayed



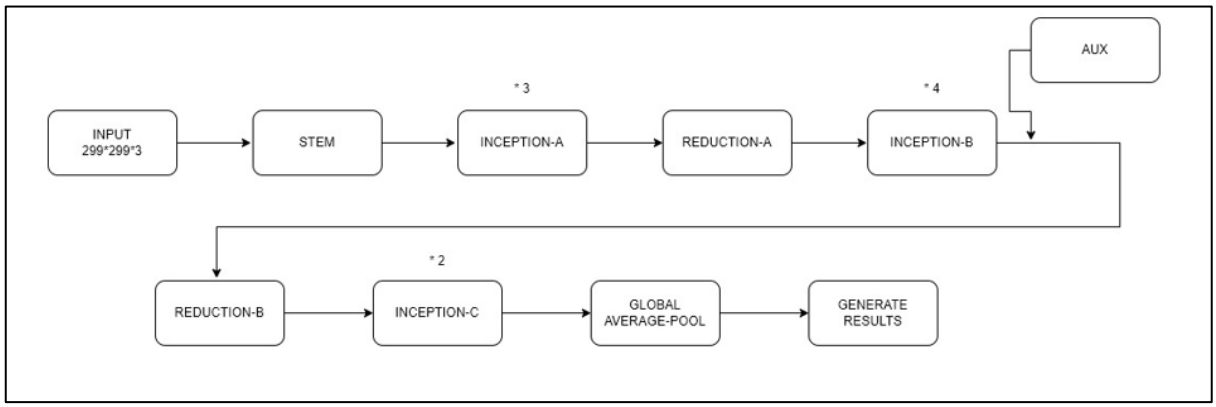
3. VGG-16:

- First, we import the FastAI module and its APIs. These APIs help in building the architecture of the VGG-16 model.
- The pretrained FastAI-VGG-16 model is imported
- Hyperparameters and other details are specified
- The model is re-trained with the preprocessed dataset
- Accuracy and other metrics are displayed



4. INCEPTION-V3

- First, we import the keras module and its APIs. These APIs help in building the architecture of the InceptionV3 model.
- After the validation split and defining number of epochs, the InceptionV3 is imported.
- Deciding if load is pre-trained with weights from Imagenet
- The base model is trained after adding some layers
- Some layers are frozen.
- Hyperparameters and other details are specified
- Accuracy and other metrics are displayed



DATASET

Images of various types of Indian architecture were collected from the internet through web scraping. The types are, Cave architecture, Chola, Hoysala, Kalinga, Kerala, Maru-Gurjara, Mughal, Rajput, Tibetan. After this, manual refinement was done for duplicates and watermarks.

These images were then further refined by removing the pictures with humans and other objects taking up space in the frame. Data augmentation was also done in an attempt to increase the accuracy of our models.

The dataset used contains 1700 images which was used for 3 models (Fast-AI VGG, Fast-AI ResNet, CNN-SVM) and another version of the same which was augmented and preprocessed again to about 5500 images that was used for 1 model (InceptionV3).

Dataset Link:

https://drive.google.com/drive/folders/158m1G4pKsc50tXUrcAjw0uXcYu-tpYNF?usp=sharing

Image Scrapper Code:

```
def get_images_from_google(wd, delay, max_images):
    def scroll_down(wd):
        wd.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(delay)

    url = "https://www.google.com/search?q=City+Palace,+Jaipur&client=firefox-b-d&channel=crow5&sxsrf=APq-WBsstJptvF4qil
    wd.get(url)


    image_urls = set()
    skips = 0
    print(max_images)
    while len(image_urls) + skips < max_images:
        print(len(image_urls) + skips)
        scroll_down(wd)
        thumbnails = wd.find_elements(By.CLASS_NAME, "Q4LuWd")
        if len(image_urls) + skips >= max_images:
            break
        for img in thumbnails[len(image_urls) + skips:max_images]:
            try:
                img.click()
                time.sleep(delay)
            except:
                continue

            images = wd.find_elements(By.CLASS_NAME, "n3VNCb")
            for image in images:
                if image.get_attribute('src') in image_urls:
                    max_images += 1
                    skips += 1
                    break

                if image.get_attribute('src') and 'http' in image.get_attribute('src'):
                    image_urls.add(image.get_attribute('src'))
                    print(f"Found {len(image_urls)}")

    return image_urls
```

RESULT

The following was obtained after running all the above mentioned algorithms on our dataset.

CNN-SVM:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
1	0.0001	0.25	4	20	0.76562	0.84313	0.671875
2	0.00008	0.25	4	20	0.73437	0.83417	0.64843
3	0.00005	0.25	4	20	0.69922	0.79144	0.57812
4	0.0001	0.1	4	20	0.76953	0.83796	0.70703
5	0.00008	0.1	4	20	0.74218	0.80769	0.65625
6	0.00005	0.1	4	20	0.69513	0.79032	0.57421
7	0.0001	0.25	6	20	0.76953	0.82222	0.72265
8	0.00008	0.25	6	20	0.74218	0.80645	0.68359
9	0.00005	0.25	6	20	0.69921	0.78048	0.62500
10	0.0001	0.1	6	20	0.76562	0.81623	0.74609
11	0.00008	0.1	6	20	0.75781	0.81081	0.70312
12	0.00005	0.1	6	20	0.71093	0.75688	0.64453

RESNET:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
1	0.001	Default	Default Resnet34 variant	6	0.902235	0.902649	0.903412
2	maxLR -> (0.001 to 0.0001)	Default	Default Resnet34 variant	6	0.821229	0.827295	0.824500
3	0.001	0.25	Default Resnet34 variant	6	0.888268	0.892145	0.889804
4	maxLR -> (0.001 to 0.0001)	0.25	Default Resnet34 variant	6	0.837989	0.838241	0.836381
5	0.001	0.1	Default Resnet34 variant	6	0.913408	0.911673	0.912555
6	maxLR -> (0.001 to 0.0001)	0.1	Default Resnet34 variant	6	0.865922	0.865039	0.862600
7	0.001	Default	Default Resnet50 variant	6	0.854749	0.856508	0.852813
8	maxLR -> (0.001 to 0.0001)	Default	Default Resnet50 variant	6	0.924581	0.925023	0.923364
9	0.001	0.25	Default Resnet50 variant	6	0.865922	0.868281	0.864022
10	maxLR -> (0.001 to 0.0001)	0.25	Default Resnet50 variant	6	0.902235	0.901211	0.899378
11	0.001	0.1	Default Resnet50 variant	6	0.863128	0.859708	0.860474
12	maxLR -> (0.001 to 0.0001)	0.1	Default Resnet50 variant	6	0.899441	0.899097	0.897183

VGG-16:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
1	0.001	Default	16	6	0.837989	0.843861	0.838468
2	maxLR -> (0.001 to 0.0001)	Default	16	6	0.905028	0.903925	0.904533
3	0.001	0.25	16	6	0.846369	0.846320	0.846595
4	maxLR -> (0.001 to 0.0001)	0.25	16	6	0.885475	0.884531	0.885507
5	0.001	0.1	16	6	0.849162	0.848724	0.846431
6	maxLR -> (0.001 to 0.0001)	0.1	16	6	0.899441	0.902293	0.901609

Inception-V3:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
1	0.0001	0.25	5	100	0.6468	0.6957	0.6180
2	0.00008	0.25	5	100	0.6414	0.6928	0.6054
3	0.00005	0.25	5	100	0.6811	0.7197	0.6198
4	0.0001	0.1	5	100	0.6757	0.7060	0.6577
5	0.00008	0.1	5	100	0.6360	0.6820	0.6144
6	0.00005	0.1	5	100	0.6559	0.6980	0.6288
7	0.0001	0.25	7	100	0.5982	0.6509	0.5441
8	0.00008	0.25	7	100	0.6144	0.6588	0.5568
9	0.00005	0.25	7	100	0.6306	0.7034	0.5640
10	0.0001	0.1	7	100	0.6414	0.6806	0.6180
11	0.00008	0.1	7	100	0.6721	0.7137	0.6468
12	0.00005	0.1	7	100	0.6613	0.7014	0.6306

DISCUSSION

From the above tables, it can be seen that Fast-AI ResNet has the best performance accuracy (92.4%) as its highest amongst the others. Closely followed by Fast-AI VGG (90.5%). Then comes CNN-SVM (76.9%) and Inception-V3 (68.1%) providing the least performance. Thus, we conclude that FastAI ResNet is the best model to use to classify architectural images- which not only depend on colour and normal image features, but also spatial and edge variations.

Best Results:

CNN-SVM:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
4	0.0001	0.1	4	20	0.76953	0.83796	0.70703

RESNET:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
8	maxLR -> (0.001 to 0.0001)	Default	Default Resnet50 variant	6	0.924581	0.925023	0.923364

VGG-16:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
2	maxLR -> (0.001 to 0.0001)	Default	16	6	0.905028	0.903925	0.904533

Inception-V3:

Variation	LR	Dropout	No. of Layers	Epochs	Accuracy	Precision	Recall
3	0.00005	0.25	5	100	0.6811	0.7197	0.6198

REFERENCES

1. Kumar, A., Bhowmick, S., Jayanthi, N., & Indu, S. (2021). Improving Landmark Recognition using Saliency detection and Feature classification. In *Digital Techniques for Heritage Presentation and Preservation* (pp. 157-175). Springer, Cham.
2. Gada, S., Mehta, V., Kanchan, K., Jain, C., & Raut, P. (2017, December). Monument recognition using deep neural networks. In *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-6). IEEE.
3. Sahay, T., Mehta, A., & Jadon, S. (2017). *Architecture classification for Indian monuments*. Technical report, University of Massachusetts Amherst.
4. Paul, A. J., Ghose, S., Aggarwal, K., Nethaji, N., Pal, S., & Purkayastha, A. D. (2021, November). Machine learning advances aiding recognition and classification of Indian monuments and landmarks. In *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)* (pp. 1-8). IEEE.
5. Sharma, S., Aggarwal, P., Bhattacharyya, A. N., & Indu, S. (2017, December). Classification of Indian monuments into architectural styles. In *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics* (pp. 540-549). Springer, Singapore.
6. Obeso, A. M., Benois-Pineau, J., Acosta, A. Á. R., & Vázquez, M. S. G. (2016). Architectural style classification of Mexican historical buildings using deep convolutional neural networks and sparse features. *Journal of Electronic Imaging*, *26*(1), 011016.
7. Xu, Z., Tao, D., Zhang, Y., Wu, J., & Tsoi, A. C. (2014, September). Architectural style classification using multinomial latent logistic regression. In *European Conference on Computer Vision* (pp. 600-615). Springer, Cham.
8. Taoufiq, S., Nagy, B., & Benedek, C. (2020). HierarchyNet: Hierarchical CNN-based urban building classification. *Remote Sensing*, *12*(22), 3794.
9. https://github.com/AKASH2907/indian_landmark_recognition
10. https://www.kaggle.com/code/atulyaatul1999/image-classification-using-svm-with-cnn-in-keras/notebook
11. https://github.com/dumitrux/architectural-style-recognition