Chapter 6

CNN-EFF: CNN based Edge Feature Fusion in Semantic Class Prediction

In chapter-4 and 5, we have studied Deep feature extraction using 'off-the-shelf' network and some relaxation labelling techniques. 'off-the-shelf' networks are pre-trained and exhibit a large number of layers. Therefore, the feature extraction is computationally expensive. Mostly, pre-trained models with transfer learning have used for semantic labelling of RGB/Gray images. Therefore, this chapter studies a customized and efficient CNN framework for semantic image parsing. The efficiency has improved by using a novel relaxation labelling based method on CNN outcome.

6.1 Introduction

Semantic image segmentation has rapidly become an important area of research in computer vision and machine learning domain. Many applications have required a robust mechanism for segmentation, such as self-driving, augmentative reality, and object recognition. Nowadays, semantic labelling has a wide variety of applications in object detection, target prediction, remote sensing images, medical image segmentation, and navigation [152]. In existing schemes such as [170], the significant challenges are to learn

CNN with minimal training samples. In past works, the authors have used unprocessed images as an input to CNN [168] that required the massive computational load to learn from such huge datasets. In this chapter, we have studied a two-step frame-work that classifies the image objects into predefined labels by using a novel CNN architecture and minimal training samples. The CNN computes the softmax likelihood probabilities of label prediction. However, The likelihood probabilities of our CNN architecture are very efficient, but it is a hard classification approach in which likelihood probabilities can be further improved [154]. Therefore, we have developed the likelihood probability by an efficient feature fusion(EFF) method. In step-1, nine-layer CNN architecture has introduced, which trains on minimal training samples and results in the likelihood probabilities of prediction. These probabilities are soft estimates derived from the hard classifier, i.e., MLP. Training data in step-1 has prepared in the form of a patch-label dataset. In step-2, we have exploited a Jacobian optimization-based label relaxation method (EFF) that has fused the local extrema as an edge prior. We have denoted the proposed approach as CNN-EFF in the paper. The CNN-EFF scheme has evaluated on two publicly available benchmark datasets as image and annotated labels. The experimental results have compared with the previously proposed state-of-art methods, and it has found that the CNN-EFF approach has improved the accuracy of semantic segmentation up to a significant gain from past processes. The classification outcome has shown that the CNN-EFF method has achieved 84.42%, 85.91%, 94.66%, 97.14%, and 98.27% accuracy for highway, house, sheep, horse rider, and horse keeper images respectively obtained from sift-flow and pascal-voc datasets. Conclusively, the proposed frame-work has out-performed the previously proposed state-of-art methods. Further, we have discussed the state-of-art formulation and experimental analysis of CNN and label relaxation approach.

6.2 Proposed Method:(CNN-EFF)

In this section, a two-step CNN-EFF method has demonstrated for efficient scene parsing. The step-1 has illustrated the CNN architecture, which has resulted in the posterior probability of semantic prediction. In step-2, we have formulated a Jacobian optimization-based label relaxation approach, that has improved the posterior probabilities by the fusion of contextual features. The label relaxation has denoted as EFF for the

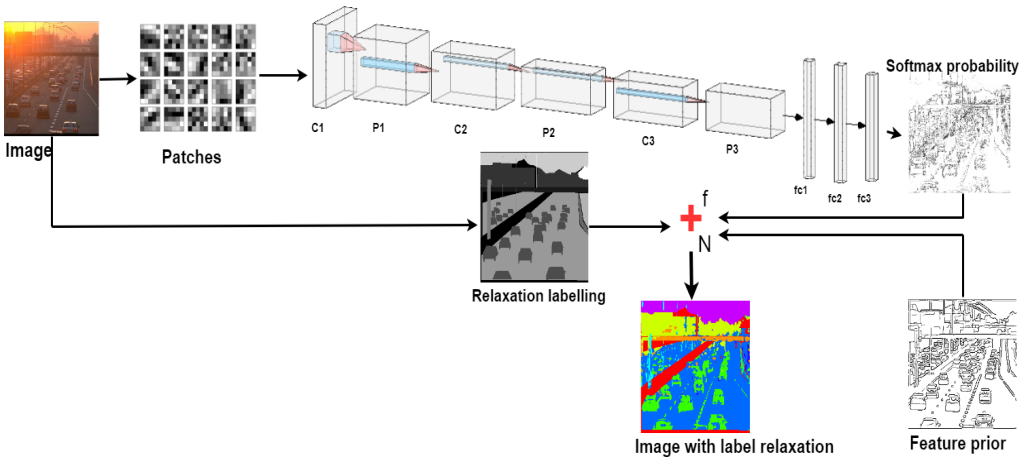


FIGURE 6.1: Process flow for CNN-EFF method

post-processing of CNN. The process flow for the proposed CNN-EFF method has shown in Figure-6.1.

6.2.1 Deep CNN based training and soft-max probability computation

In this section, we have introduced a CNN architecture which yields the soft-max probabilities for each pixel in one of the available classes. The CNN model contains three pairs of convolution-pooling (conv-pool) layers and three fully connected layers. The conv-pool pairs have extracted the main features from the patch set and passed into the fully connected layers. For data input, the CNN model has extracted the patch-label sets and predicts pixel probability that belongs to a labelled class. The prediction probability has obtained by the logits of the soft-max layer. Let the patch label dataset has size $p \times p \times 3$ along with their image patch labels. Further architecture has demonstrated as below:

6.2.1.1 First convolution and maximum pooling Layer(conv-1 and pool-1)

In Conv-1 layer, our kernel size is 3×3 and number of such filter are 300. These filters have applied to our input patch label data-set. In this layer, let the output feature has the size $f_1 \times f_1 \times 3$ where $f_1 = (p-2)$. P is the patch size of our generated data-set. The output of

this layer has computed as:

$$F_1 = max(0, w_1 * x + b_1) \tag{6.1}$$

300 features of size= $3 \times 3 \times 3$ has applied to obtain w_1. The output of conv-1 layer is: 300 features of $(f_1 \times f_1)$ size. These features have passed into pool-1 layer where the pooling has performed on convoluted features. The pooling constrain is $f_2 \times f_2 \times 300$ and $f_2 = \left\lceil \frac{f_1}{2} \right\rceil$. The pool-1 features then get into the next conv-2 layer.

6.2.1.2 Second convolution and maximum pooling Layer(conv-2 and pool-2)

The pool-1 features obtained from the previous layer have passed into the conv-2 layer. In this layer, the kernel has reduced to 200, and its size is 3×3. The conv-2 layer has transformed the features into a new size of $f_3 \times f_3 \times 200$ whereas $f_3 = f_2 - 2$. The output of this layer is:

$$F_3 = max(0, w3 * F_2 + b3) \tag{6.2}$$

The conv-2 features has fed into the pool-2 layer which is second pooling layer. The kernel size of pool-2 layer is two. The features have the size of $f_4 \times f_4 \times 200$ where $f_4 = \left\lceil \frac{f_3}{2} \right\rceil$. The outcome of pool-2 layer has shown as F_4

6.2.1.3 Third Convolution and pooling Layer(conv-3 and pool-3)

In conv-3 layer, the kernel has the size of 2 and filter count has $f_5 \times f_5 \times 200$ where $f_5 = f_4 - 1$. The output of this layer is:

$$F_5 = max(0, w5 * F_4 + b5) \tag{6.3}$$

conv-3 features have get into the pool-3 layer of kernel size two. The features are $f_6 \times f_6 \times 200$ where $f_6 = \left\lceil \frac{f_5}{2} \right\rceil$. We have flatten the output and passed into fully connected layers. The output of pool-3 layer has denoted as F_6.

6.2.1.4 Data in fully connected (fc) Layers

We have demonstrated the evaluation of previous features in the fully connected (FC) layer. Let the features of FC layers are (F_7, F_8, F_9). The weights and biases assumed as (w_7, w_8, w_9) and (b_7, b_8, b_9). Therefore the features of this layer are:

$$F_i = \phi(w_i * F_{i-1} + b_i) \tag{6.4}$$

the layer i=(7,8)
Features of 9^{th} layer is:

$$F_9 = \phi(w_8 * F_8 + b_9) \tag{6.5}$$

weights and biases can be denoted as:

$$w = sgdm\{(w_1, w_3, w_5, w_6, w_7, w_8, w_9)\} \tag{6.6}$$

$$b = sgdm\{(b_1, b_3, b_5, b_6, b_7, b_8, b_9)\} \tag{6.7}$$

In above equation the (w_1, w_3, w_5) and (b_1, b_3, b_5) are the weights and bias update of convolution layer. Next f_9 has fed into the soft-max layer to achieve the semantic pixel probability for labelled classes. Suppose that the $x_i^{(w,b)}$ is the soft-max prediction of classes for image pixels. The Classes can be obtained as:

$$y_i^{w,b} = \underset{i \varepsilon K}{argmax}\ (x_i^{(w,b)}) \tag{6.8}$$

K is number of classes and y_i is the classes for $y_i^{(w,b)}$ which is the class prediction. Hence, the loss can be defined as;

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} E(y_i, y_i^{(w,b)}) \tag{6.9}$$

Let consider the loss function as $L(\theta)$ which has to be optimized using sgdm method. The weights and biases can be obtained in i_{th} epoch as:

$$w_{i+1} = w_i + \eta \times \left[\frac{\partial L(\theta)}{\partial w} \right]_{w_i} \tag{6.10}$$

$$b_{i+1} = b_i + \eta \times \left[\frac{\partial L(\theta)}{\partial b}\right]_{b_i} \tag{6.11}$$

The back propagation error method has applied to find the gradient of w and b after 1000 epoch. The learning rate is 0.01. In the next section, we have demonstrated the CNN-EFF approach for scene relaxation labelling using Jacobian optimization. This scheme has used for relaxation on the probabilities obtained in this section.

6.2.2 Edge Feature Fusion based post processing(EFF-Jacobian optimization)

In this section, we have studied a Jacobian optimization-based label relaxation scheme, which reduced the spatial and spectral distortion in posterior probabilities of CNN. We have applied the total variation(TV) based contextual fusion method in convex optimization. In various works, the TV regularization method has significantly reduced the noise in RGB images. In this work, we have used a similar objective function as in past methods but with significant modifications. We have included the shape prior that has fused the effect of spatial neighbourhood extremes obtained from a novel feature extraction method in the posterior probability. In past references [137, 178, 175], the authors have applied the constrained optimization based objective function on the probabilistic values by using spatial knowledge. We have adopted the concept of relaxation with edge prior to [129], in which the author has included the edge values in their objective function to improve the prediction results. We have restructured and modified the objective function in such a way that it takes edge prior, as described in Algorithm-5 and equation-6.13, and optimize this function by using Jacobian optimization. In Jacobian optimization, we have first computed the total variation with the edge prior to each pixel. In the end, we have included this information with probabilistic values. Whereas in the previous paper, [129], the authors have added the TV values iteratively right after it has computed. Let the Soft-Max posterior probability is p, and x is the optimized probabilistic values that have to compute. Let the target classes C_i has predicted by maximizing the posterior x as $[C_i = \underset{c}{Argmax}\ \phi_i^c]$ where $i\ \varepsilon\ (number\ of\ pixels)$. The relaxation prior has modeled as:

$$\phi(x) = \sum_{|i-j|<N} ||x_i - x_j||_1 \oplus_N^j f_j \tag{6.12}$$

where $\oplus_N^j$ is neighbourhood multiplier of j neighbours where $j\ \varepsilon\ N$ and N is the number of neighbours of a particular pixel. x_i is i^{th} value of x for pixel i. $||\{\}||_1$ is the L_1 norm, $|i-j|$ is the neighbourhood between i and j pixels and N is number of neighbours. We have obtained the likelihood posterior p from the CNN model and transformed them into posterior x as:

$$\hat{x} = \underset{x}{argmin}\ 0.5\ \times\ ||x-p||^2 + \mu\phi(x) \tag{6.13}$$

The first part of the above equation controls the spectral probability, and the second part has denoted the total variation(TV) based relaxation. μ is the regularizer to control spectral and spatial knowledge. We have applied a Jacobian optimization for the solution of the objective function in algorithm-1. The optimization process requires five inputs that are edge features (f), number of iterations (iter), regularizer(μ), nearest neighbourhood(NN), posterior probability(p). The Jacobian optimization has started with the probabilistic output of our CNN architecture. Let the posteriors, edge prior, nearest neighbourhood, and regularizer are p, f, NN, and μ, respectively obtained form CNN as:

$$p \leftarrow Soft-Max\ Probability$$

$$\mu \leftarrow Regularizer$$

Let the RGB image is I and the gradient in x direction is I_x:

$$\triangle I_x = I(x+1,y) - I(x,y) \tag{6.14}$$

Similarly the gradient in y direction is I_y:

$$\triangle I_y = I(x,y+1) - I(x,y) \tag{6.15}$$

$\triangle I_x$ and $\triangle I_x$ are the per pixel derivative of image I. Since the image is RGB having 3 bands, therefore, mathematically $\triangle I_x$ and $\triangle I_y$ can be computed as in Algorithm-3:

$$[Row, Col, Ch] = size(I) \tag{6.16}$$

We can compute the edge prior(f) as the final form of our gradient as in Algorithm-3 and Equation-6.17:

$$f = \frac{1}{N^2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sqrt{\frac{\triangle I_1^2 + \triangle I_2^2}{2}} \tag{6.17}$$

Algorithm 3 Compute $\triangle I_x$ and $\triangle I_y$

$Inputs \leftarrow I, Row, Col, Ch$
for $[k \leftarrow 1 : Ch]$ **do**
  for $i \leftarrow 1 : Row$ **do**
    for $j \leftarrow 1 : Col - 1$ **do**
      $\triangle I_x(i,j,k) \leftarrow abs(I(i,j+1,k) - I(i,j-1,k));$
    end for
  end for
end for
for $[k \leftarrow 1 : Ch]$ **do**
  for $i \leftarrow 1 : Row - 1$ **do**
    for $j \leftarrow 1 : Col$ **do**
      $\triangle I_y(i,j,k) \leftarrow abs(I(i,j+1,k) - I(i,j-1,k));$
    end for
  end for
end for
$\triangle I_1 \leftarrow sum(\triangle I_x, 3)$
$\triangle I_2 \leftarrow sum(\triangle I_y, 3)$

Along with the gradient value, we require spatial neighbourhood points of each pixel on which the gradient information has fused. Eight nearest neighbours have taken for each pixel of the image, but some other neighbourhood proximity like 4 or 16 can also be considered. More spatial neighbours can increase the complexity of the fusion process. The image I have padded with zeros for corner pixel neighbourhood. Uniform padding with one row and one column has performed as:

$$I_{pad} = pad\_Image(I, [1,\ 1], 0); \tag{6.18}$$

Equation-6.18 denotes the zero padding in each side of row and column of image I. Now a container *NN* for nearest neighbours is initialized as:

$$NN = zeros(Row * Col, 9); \tag{6.19}$$

Let the counter for the pixel index is C=0. In 8-neighborhood, we have to remove the column representing the 5^{th} pixels at the end because it is the same central pixel for which neighbourhood has calculated. The neighbourhood order has denoted in Figure-6.2. The algorithm-4 denotes the process of finding the 8-nearest neighbours in an image grid.

Next, we have applied a Jacobian optimization based techniques for the fusion of

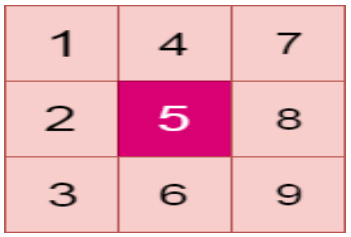


FIGURE 6.2: Pixel-ordering in 8-Neighbour computation for pixels-5(central pixel)

Algorithm 4 Compute Nearest Neighbours(NN)

$Inputs \leftarrow I_{pad}, C = 0, NN, Row, Col, Ch$
for $[i \leftarrow 2 : Col + 1]$ **do**
  for $j \leftarrow 2 : Row + 1$ **do**
    $C \leftarrow C + 1$ ▷ Neighbourhood Search
    $nbrs \leftarrow I_{pad}(Row - 1 : Row + 1, Col - 1 : Col + 1)$ ▷ Finding the 8 neighbours
    $nbrs \leftarrow nbrs(:)'$
    $NN(C,:) \leftarrow nbrs;$ ▷ pixel-Neighbours are stored in Row-Column format
  end for
end for
$NN(:,5) = []$ ▷ Column-5 can be removed

neighbourhood values of edge before the CNN based estimates. The convergence criterion is the number of iterations(iter) in this solution. As discussed above, the CNN softmax probability is p, and iterations are iter, regularizer is μ. We have computed the edge prior 'f' and nearest neighbourhood as 'NN' in the algorithms-3 and 4, respectively. NN is a $N \times 8$ matrix of 8 neighbours for N pixels. The neighbourhood information fusion has computed as Equation-6.20:

$$fusion = \sum_{i=1,j=1}^{Classes(C),Pixels(N)} [p(i,NN(j,:))\ Kron\ f(NN(j,:))] \tag{6.20}$$

Kron is the kronecker product in equation-20 between the edge prior and pixel values according to their nearest neighbours values. The optimized pixel values are reported in Equation-6.21 as:

$$x_{new} = \frac{[0.5 \times p + \mu \times fusion]}{0.5 + \mu \times \sum_{i,j}^{N} f(NN(j,:)} \tag{6.21}$$

The x_{new} values obtained in the above algorithm is the optimized posterior, which is the improved probabilities. We have received the absolute error, which has transformed the problem into a convex problem and swiftly converged it. The complete process has

demonstrated in algorithm-5. The decline in the backward difference in the error has been shown the convex convergence of optimization. In subsequent sections, We have experimentally analyse the significance of our CNN-EFF method.

Algorithm 5 EFF based Jacbian optimization

$Inputs \leftarrow p, iter, \mu, NN, f$
$Output \leftarrow x_{new}$
$v \leftarrow imageTOvector(f)$
$[class, samples] \leftarrow size(p)$
$p1 \leftarrow p$
for $[t = 1 : iter]$ **do**
  $p2 = p$
  for $j = 1 : samples$ **do**
    $p3 \leftarrow p1(all, j)$
    $p4 \leftarrow p(all, j)$
    $temp \leftarrow NN(j, all)$
    $\sigma \leftarrow sum(f(temp))$
    for $k \leftarrow 1 : class$ **do** $spatial_{features} = sum(p1(k, temp)). \times f(temp)$
    end for
    $x_{new} = \frac{[0.5 \times p + \mu \times spatial_{features}]}{0.5 + \mu \times \sigma}$
    $p1 \leftarrow x_{new}$
    $p2 \leftarrow p1$
  end for
end for

6.3 Experimental Result Analysis

In this section, we have conducted the image-based experiments on two publicly available labelled data-sets .i.e PASCAL -visual object class(pascal-voc) and sift-flow data-sets. We have performed the preprocessing of labelled samples so that each labelled image contained the labels in lexicographic order. Subsequently, we have performed the two-stage CNN-EFF method for the semantic classification of pixel labels in the images. Finally, we have compared the prediction accuracy and computation time with [200, 180, 158].

6.3.1 Data-set details

We have evaluated the proposed CNN-EFF method on sift-flow and pascal-voc data-sets. Sift-flow is a 672 MB data-set that contains two fields: Images and Labels. Images field contains the RGB images of size $256 \times 256 \times 3$ and 'Label' field contains the corresponding labelled image with size 256×256. Each image in the sift-flow data-set has a similar size, whereas the pascal-voc data-set contains the images in different sizes. Since we have proposed a patch-based CNN training method, therefore, the initial size of images does not matter. We have not taken all the images from the sift-flow and pascal-voc data-set. We have selected some specific images for patch-based training, which has used in [200], a recently proposed work, for comparative purpose. In the preprocessing step, we have replaced the class labels in the increasing order of class labels and generated the class-wise patches of image and labels. The advantage of class-wise patches is that the class label of each patch has determined by the corresponding patch label's central pixel. In the next section, we have reported some metrics, training characteristics, and performance of CNN-EFF scheme.

6.3.2 Evaluation Metrics

The evaluation has performed based on some standard metrics .i.e OA(Overall Accuracy), AA(Average accuracy), IOU (Intersection over union), and computational time. These metrics have calculated from the confusion matrix of prediction and ground-truth image. Let $p_{i,j}$ is the number of pixels in an image that genuinely belongs to class i but have predicted and labelled in class j, where c is the total number of classes. Therefore, we can write the $p_{i,j}$ as:

$$p_{i,j} = \sum_{(X,G)\varepsilon D} \sum_{m,n} [g^{m,n} = i \cap y^{m,n} = j] \tag{6.22}$$

Here X, G is the image, ground-truth and D is validation set.
The true positives for class i is:

$$Tp_i = \sum_{j=1}^{c} p_{i,j} \tag{6.23}$$

The true positives for class j is:

$$Tp_j = \sum_{i=1}^{c} p_{i,j} \tag{6.24}$$

The total number of pixels:

$$N = \sum_{i=1}^{c} \sum_{j=1}^{c} p_{i,j} \tag{6.25}$$

The overall accuracy is:

$$OA = \frac{\sum_{i=1}^{c} p_{i,i}}{N} \tag{6.26}$$

The average accuracy is:

$$AA = \frac{\sum_{i=1}^{c} \frac{c_{i,i}}{n_i}}{c} \tag{6.27}$$

Overall accuracy and average accuracy has calculated by the confusion matrix. In computer vision and semantic labelling problems, the IOU(intersection over union) or Jaccard index has also been computed to obtain the accuracy which is not dependent on data distribution. The IOU is:

$$IOU = \frac{1}{c} \sum_{i=1}^{c} \frac{c_{i,i}}{Tp1_i + Tp2_i - c_{i,i}} \tag{6.28}$$

All these indices have evaluated the performance of the proposed CNN-EFF method in a better sense. These indices are sufficient to analyse the prediction scenario in supervised classification methods. However, some other matrices such as precision, recall. F1 score can also be derived from confusion metrics.

6.3.3 Experiment on Sift-Flow dataset

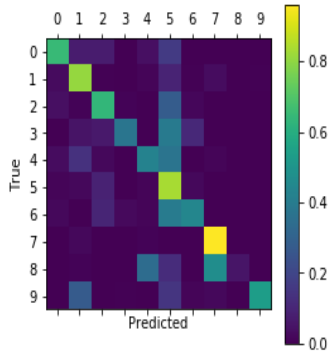
In this section, we have used the highway and house image from the sift-flow dataset. The sift-flow dataset is containing more than 7000 images with 33 labelled classes. But the highway and house image consist of the images having twelve and ten classes out of them.

6.3.3.1 Highway dataset

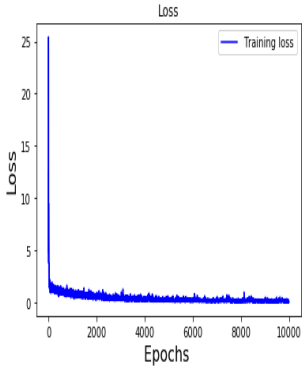
For highway data-set, we have first prepared the class-wise patch-label data-set to train the CNN model. The training has performed on the proposed CNN, having three pairs of convolution-pooling layers and three fully connected layers. In such a way, CNN has

TABLE 6.1: Testing results for sift-flow dataset

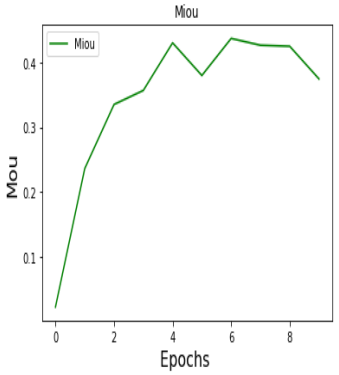
Classes	sift-flow(highway)		sift-flow(house)	
	CNN	CNN-EFF	CNN	CNN-EFF
c1	54.41	75.59	79.33	81.30
c2	74.31	85.84	37.22	64.66
c3	51.71	71.29	47.19	64.62
c4	50.92	89.96	74.92	84.37
c5	24.25	42.22	81.29	83.50
c6	74.43	84.00	45.95	60.11
c7	36.76	62.12	96.84	98.73
c8	86.34	93.83	51.57	68.99
c9	52.38	62.50	67.57	83.95
c10	48.58	63.38	27.78	28.95
c11	-	-	47.04	58.33
c12	-	-	45.47	67.37
OA	69.11	84.42	80.75	85.91
AA	50.54	76.73	50.16	79.02
IOU	-	.8822	-	97.83
Time(EFF)	-	16.91(sec)	-	20.08



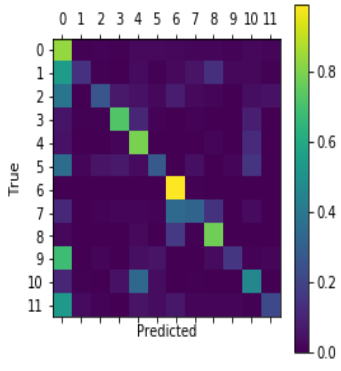
(A) Confusion matrix-highway dataset



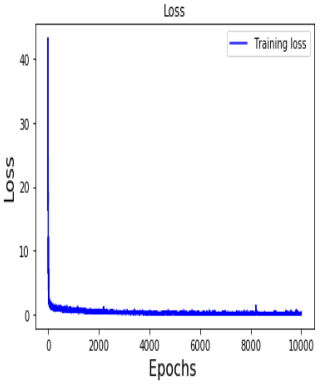
(B) Training Loss-highway dataset



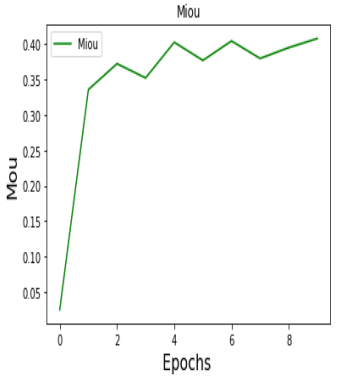
(C) MIOU-highway dataset



(D) Confusion matrix-house dataset

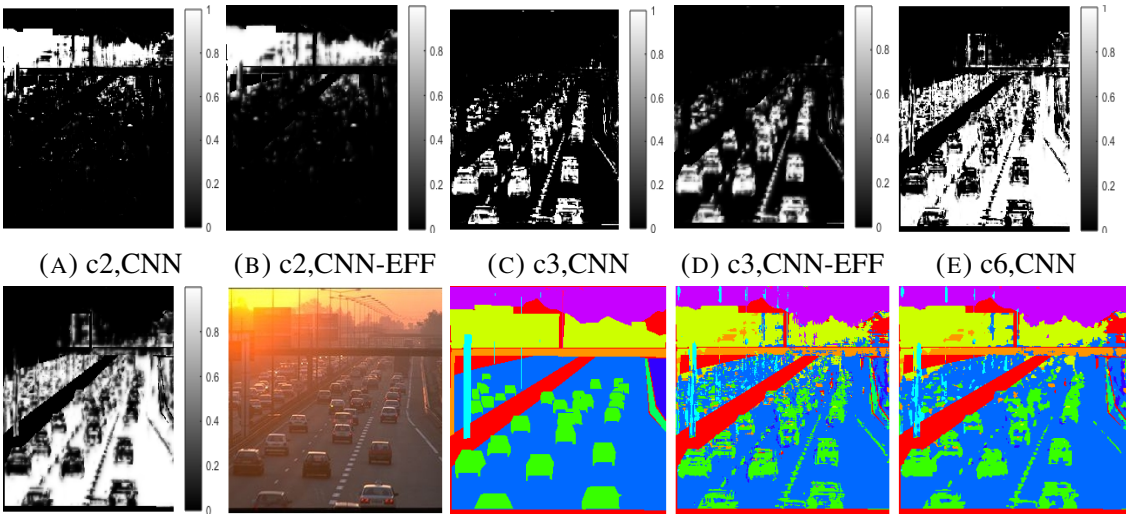


(E) Training Loss-house dataset



(F) MIOU-house dataset

FIGURE 6.3: Confusion matrix, Training Loss, MIOU of CNN-EFF method for highway and house image from sift-flow dataset

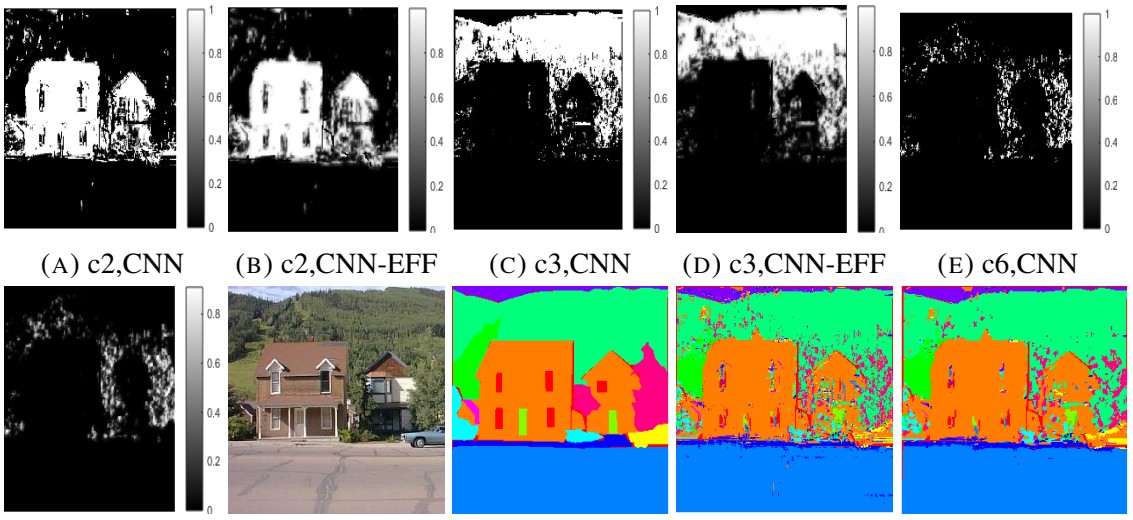


(A) c2,CNN (B) c2,CNN-EFF (C) c3,CNN (D) c3,CNN-EFF (E) c6,CNN

(F) c6,CNN-EFF (G) Highway (H) Ground truth (I) CNN,69.11 (J) CNN-EFF,84.42

FIGURE 6.4: Probability images and classification results for Highway data(sift-flow)

designed the input of fully connected layers and has passed into the softmax layer, which has provided the class-wise probability of semantic prediction. We have used 1% training patches to learn the CNN and complete patch set for predictive testing. CNN testing has performed after every 1000 epochs. Figure-6.3(a),(b), and (c) have shown the confusion matrix, training loss, and mean intersection over union(MIOU) for CNN training. It has observed that training loss has continuously decreased, and MIOU has increased up to 0.5 in figure-6.3(b) and 6.3(e). The elapsed time is 1282.30 for the training of highway data-set. After CNN training, we have obtained the prediction probability of pixels. Further, we have performed post-processing of probabilities by our Jacobian optimization-based EFF(edge feature fusion) method. The EFF method has significantly improved the probabilities of prediction, as shown in table-6.1. In table-6.1, it has been found that the overall accuracy and average accuracy has got a significant increase of 15% for highway data-set. Class-wise accuracy for each of ten classes has greatly improved on applying EFF optimization. Class c1, c3, c4, c5, and c7 classes have reported the massive increase in accuracy up to 21%, 20%, 39%, 20%, and 26% respectively. The time taken for EFF is 16.91 seconds. Figure-6.4(a) to (f) has shown the class probability images of class c2, c3, and c6 for CNN and CNN-EFF methods. Figure-6.4(g) is the labelled ground truth, and figure-6.4(i) and 6.4(j) are the prediction results of CNN and CNN-EFF methods. The outcomes of CNN and CNN-EFF methods have shown that the proposed methods are state-of-art in the domain of semantic predictions.



(A) c2,CNN (B) c2,CNN-EFF (C) c3,CNN (D) c3,CNN-EFF (E) c6,CNN

(F) c6,CNN-EFF (G) House image (H) Ground truth (I) CNN,80.75 (J) CNN-EFF,85.91

FIGURE 6.5: Probability images and classification results for House data(siftflow)

6.3.3.2 House dataset

In this section, the evaluation matrices for CNN and CNN-EFF have discussed. In CNN training, the sample size and other parameters are the same as in previous sections. The testing results have reported after every 1000 epochs during the CNN training. The figure-6.3(d), (e), and (f) are depicting the confusion matrix, training loss, and MIOU for testing on sample patches. Training loss has decreased, and MIOU has increased after each iteration. The elapsed time is 1416.65 for the training of the house dataset. The accuracy, IOU, and computation time for house datasets have reported in Table-6.1. We can notice that overall accuracy has got an increase of 5%, and average accuracy has increased up to 29% from CNN to CNN-EFF method. The IOU is 0.9783, and the computation time of EFF(post-processing) is 20.08 second. Classes c2, c3, c4, c6, c8, c9, c12 has achieved the remarkable increase in accuracy as 27%, 17%, 10%, 15%, 17%, 16%, and 22% respectively. Figure-6.5(a) to 6.5(f) has shown the class-wise prediction, and figure-6.5(g), 6.5(h) are the original image and ground truth. The figure-6.5(i) and 6.5(j) have reported the prediction results obtained from CNN and CNN-EFF methods. The evaluation matrices obtained from CNN and post-processing based CNN-EFF methods have found to be the state-of-art for the house dataset of sift-flow.

TABLE 6.2: performance of CNN-EFF and CNN model on pascal-voc dataset

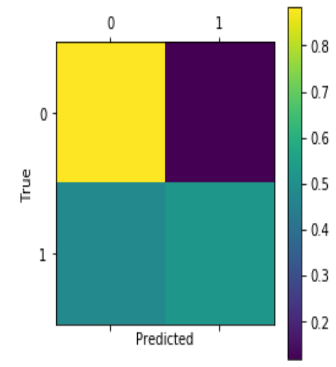
Classes	pascal-voc(sheep)		pascal-voc(Horse-rider)		pascal-voc(Horse-keeper)	
	CNN	CNN-EFF	CNN	CNN-EFF	CNN	CNN-EFF
c1	100.0	100.0	100.0	100.0	100	100
c2	87.11	86.87	80.86	84.88	96.06	96.64
c3	-	-	91.75	92.03	-	-
OA	93.76	94.66	96.59	97.14	97.74	98.27
AA	93.55	93.43	90.87	92.31	98.03	98.32
IOU	-	1	-	1	-	1
Time(EFF)	-	13.065(sec)	-	20.64	-	16.22

6.3.4 Experiment on Pascal-Voc dataset

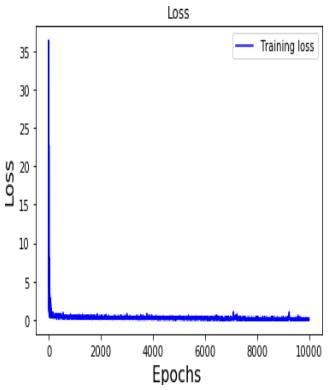
The pascal-voc dataset contains 11530 images, which have 27450 labelled objects and 6929 clusters. Pascal-voc dataset exhibits 20 overall classes in which all images have labelled. We have introduced a CNN-EFF method that trains the patch label pairs in our CNN. Therefore, we have adapted the sheep, horse rider, and horse keeper images, which contains particular classes such as sheep, horse, and man. Images in pascal-voc dataset are not similar in size and contain fewer classes per image than the sift-flow dataset. Sheep, horse rider, and horse keeper images have 2, 3, and 2 mutually exclusive classes in ground-truth.

6.3.4.1 sheep dataset

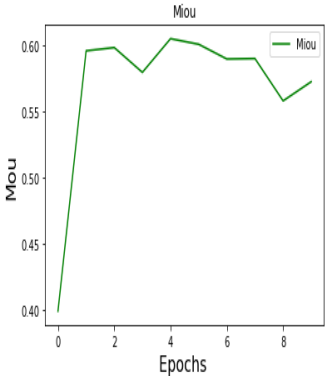
The sheep dataset contains two classes in ground-truth, and CNN parameters are similar to the sift-flow datasets of the previous section. In CNN training, we have taken a 1% patch label dataset for training and a complete dataset for testing. Figure-6.6(a), (b), (c) are the confusion matrix, training loss, and MIOU for patch testing in sheep dataset of pascal-voc. Training loss is sharply decreasing, and MIOU is sharply increasing since the number of classes is relatively less in the sheep dataset. The elapsed time for training is 1551.80 seconds. Table-2 has shown the accuracy and computation time for the proposed CNN and CNN-EFF method. Overall accuracy is increasing 1%, and the average accuracy is increasing 5% from CNN to CNN-EFF methods for sheep dataset. On measuring class-wise accuracy, in class c1, the CNN-EFF process has reported a decrease of 1%, but in class c2, it has reported an increase of 16%. Therefore, after EFF based post-processing, our CNN-EFF method has improved the performance of semantic prediction. The time



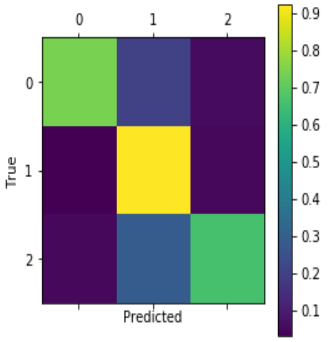
(A) Confusion matrix- sheep dataset



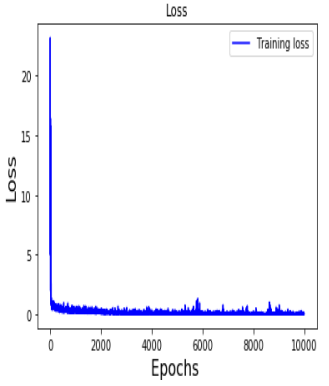
(B) Training Loss- sheep dataset



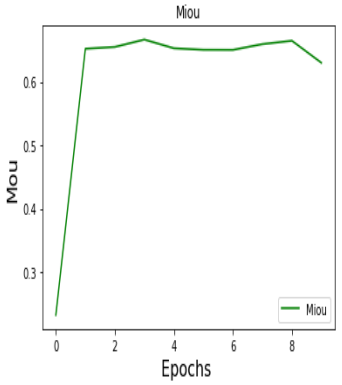
(C) MIOU- sheep dataset



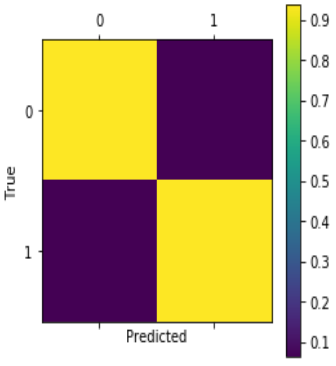
(D) Confusion matrix- horse rider dataset



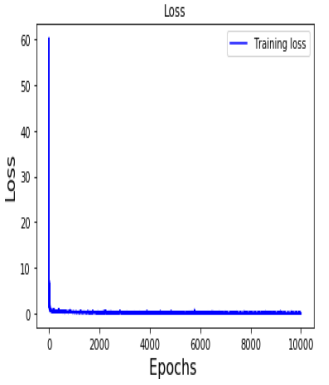
(E) Training Loss-horse rider dataset



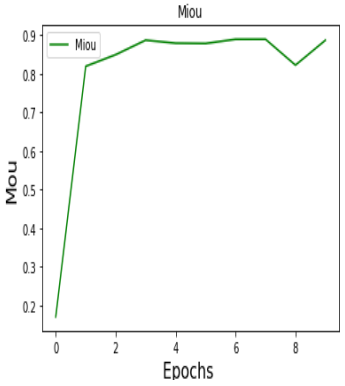
(F) MIOU- horse rider dataset



(G) Confusion matrix-horse keeper dataset

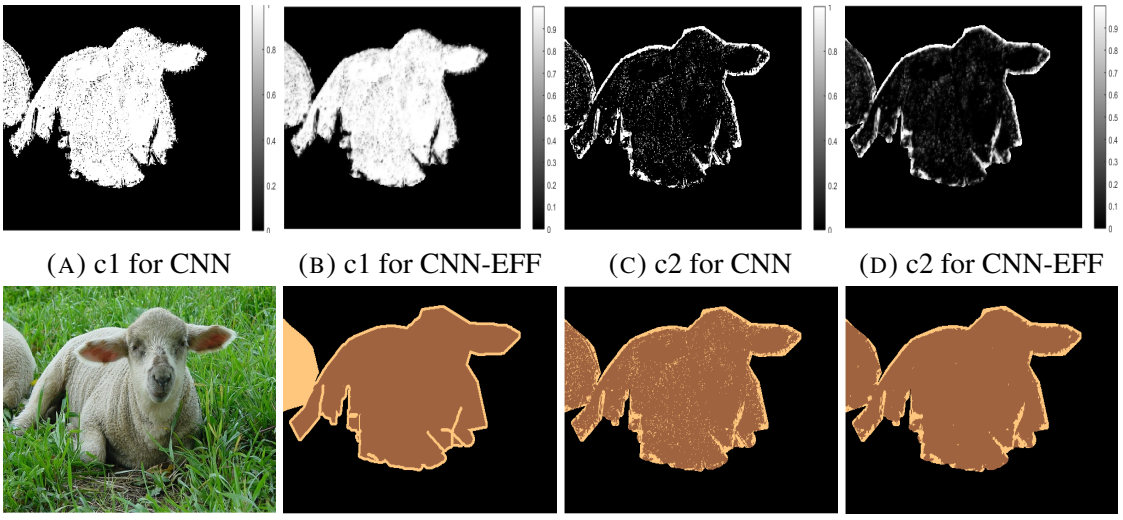


(H) Training Loss-horse keeper dataset



(I) MIOU-horse keeper dataset

FIGURE 6.6: Confusion matrix, Training Loss, MIOU of CNN-EFF method for sheep, horse rider and horse keeper image from pascal-voc dataset



(A) c1 for CNN (B) c1 for CNN-EFF (C) c2 for CNN (D) c2 for CNN-EFF

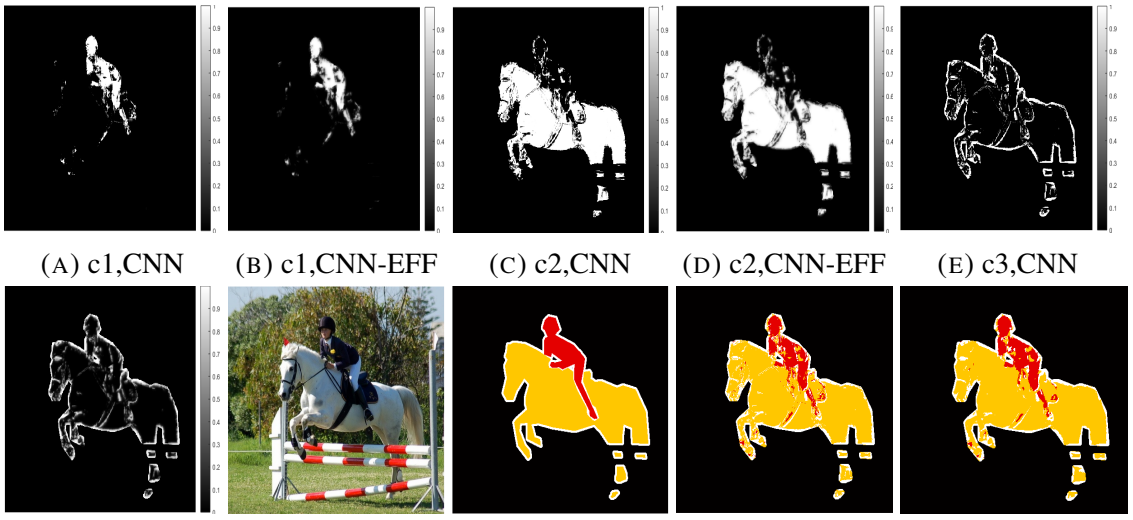
(E) sheep image (F) ground truth (G) CNN,93.76 (H) CNN-EFF,94.66

FIGURE 6.7: Probability images and classification results for sheep data(pascal-voc)

elapse for CNN-EFF methods is 13.065 seconds. The number of classes is less; therefore, the IOU is one in this case. Figure-6.7(a) to 6.7(d) has shown the class-wise prediction images for the sheep dataset. Figure-6.7(e), (f), (g), (h) are the original images, ground truth, CNN, and CNN-EFF predictions, respectively. The proposed CNN-EFF method is achieving remarkable accuracy in this case, which acknowledges the CNN-EFF approach as a state-of-art.

6.3.4.2 Horse rider dataset

This section has evaluated the CNN and CNN-EFF schemes on horse rider images of pascal-voc dataset. This image contains three class labels in its ground truth. 1% patches have taken for CNN training, and other parameters are similar to past sections. Figure-6.6(d), (e), (f) have represented the confusion matrix, training loss, and MIOU for testing patches. Training loss has a gradual decrease, and MIOU has increased in CNN testing after each epoch. In the second column of Table-6.2, we can observe that the overall accuracy has increased by 1%, and average accuracy has increased 3% from CNN to CNN-EFF method, respectively. The time elapse is 20.64 seconds for the horse rider dataset. In this data, it has found that the EFF method has improved that class accuracy by 4%, 1%, and 8% for class c1, c2, and c3, respectively, by reducing the spatial noise in probabilistic prediction. Figure-6.8(a) to 6.8(f) have shown the class-wise prediction probability image for horse



(A) c1,CNN (B) c1,CNN-EFF (C) c2,CNN (D) c2,CNN-EFF (E) c3,CNN

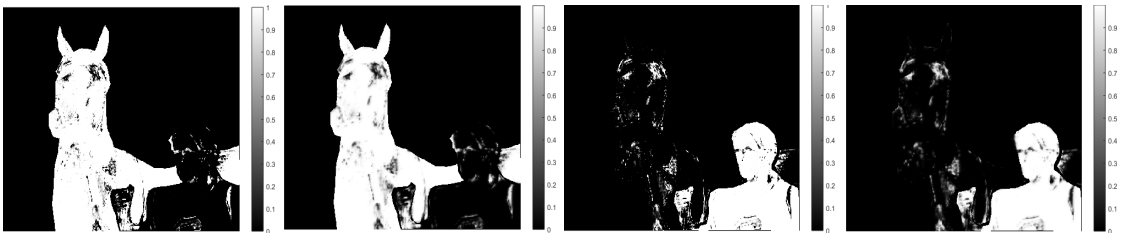
(F) c3,CNN-EFF (G) Horse-rider (H) Ground truth (I) CNN,96.59 (J) CNN-EFF,97.14

FIGURE 6.8: Probability images and classification results for Horse-rider data(pascal-voc)

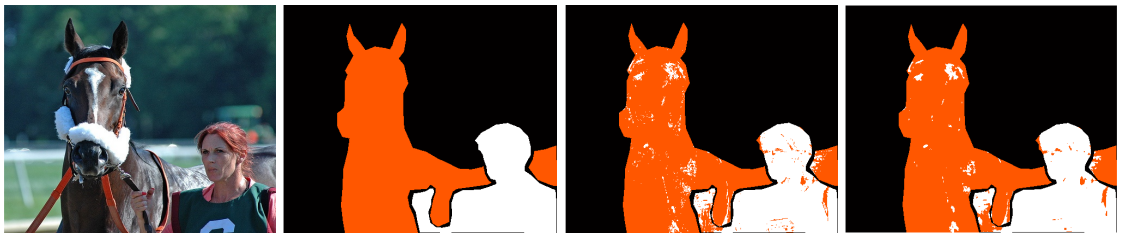
rider image. Figure-6.8(g) and 6.8(h) are the real and ground truth images. The figure-6.8(i) and 6.8(j) are the prediction results achieved for CNN and CNN-EFF method. Accuracy and time metrics have shown that the proposed CNN-EFF method is a framework that has improved the prediction accuracy significantly for both class-wise and overall way.

6.3.4.3 Horse keeper dataset

In this section, we have performed the proposed framework for the horse keeper image of pascal-voc image collection. By preprocessing the dataset, we have organized the ground truth in two mutually exclusive classes. Similar to the previous section-6.3.4.2, we have generated the patch label dataset from the horse keeper image and split the patch label dataset as 1% training samples for CNN. Figure-6.6(g), (h), and (i) have denoted the confusion matrix, training loss, and MIOU for CNN training. The training loss is continuously decreasing, and MIOU is increasing after each iteration. Table-6.2 has denoted the evaluation matrices in the third column. In the horse keeper image, the overall accuracy of the CNN-EFF method is 1% higher than CNN methods. Average accuracy has reported a small increase .i.e 0.29% and total elapsed time are 16.22 seconds. In the class-wise accuracy, class c1 has achieved maximum efficiency for both CNN and CNN-EFF methods. In class c2, we have observed an increase of 0.58% from CNN to CNN-EFF schemes.



(A) c1 for CNN (B) c1 for CNN-EFF (C) c2 for CNN (D) c2 for CNN-EFF



(E) Horse keeper image (F) Ground truth (G) CNN(97.74%) (H) CNN-EFF(98.27%)

FIGURE 6.9: Probability images and classification results for horse-keeper data(pascal-voc)

Figure-6.9(a) to (d) have denoted the class-wise probability for CNN. Figure-6.9(e) and (f) are the original image and ground truth for horse keeper dataset. Figure-6.9(g) and 6.9(h) are the prediction results for CNN and CNN-EFF methods. Further, we can deduce that our CNN-EFF approach is increasing the class-wise and overall accuracy from CNN to the CNN-EFF method. For pascal-voc and sift-flow datasets, our proposed CNN-EFF has significantly increased the accuracy by consuming the minimal computation time for the EFF based post-processing scheme.

6.3.5 Comparative Analysis

6.3.5.1 Comparison with other CNN based approaches

In this section, we have compared the results obtained from CNN and CNN-EFF method with past CNN based methods. Primarily we have shown the comparison with the methods recently proposed in [200]. In the referred work, the authors have proposed an ensemble method that combines the student model's logits and the ensemble of fully

connected layers. This method has proved to be the state-of-art for semantic target detection in sift-flow and pascal-voc data-sets. Table-6.3 has shown the overall accuracy achieved by referenced methods for different class objects. The table-6.3 indicates that the car object has achieved up to 69.60%, 65.5%, and 71.20% accuracy in single-FCN-8, compressed FCN, and ensemble FCN models. In our prediction results, class c3 represents the car prediction, which is 71.29% accuracy in CNN-EFF methods, as shown in table-6.1. Therefore, it can be noted that the CNN-EFF process is achieving higher accuracies than past methods. The house data-set in the sift-flow image, class c4 in table-6.1, has made 84.37% prediction accuracy in the CNN-EFF method. In house data-set, the prediction accuracy is 1-2% higher than single FCN-8 and compressed FCN-8 method but has performed equivalently to the ensemble method. In sheep data-set, the CNN and CNN-EFF method is achieving 87.11% and 86.87% accuracy while previously proposed FCN-8, compressed FCN-8, and ensemble FCN methods have 70.70%, 71.00%, and 77.40 % accuracy respectively. Therefore, our proposed CNN-EFF method is achieving 17%, 16%, and 10% higher accuracy than past methods. In table-6.3, it has observer that in horse rider data-set our CNN-EFF approach is achieving 84.88%, 92.03% accuracy for horse and man object which is 21%, 20%, and 14% higher for class horse and 12%, 12%, and 10% higher for the class person than FCN-8, compressed FCN, and ensemble FCN method respectively. In horse keeper image, the CNN-EFF is reporting 100% and 96.64% accuracy for class horse and person which is 37%, 36%, and 30% greater for horse and 16%, 16% and 14% higher for person object than FCN-8 compressed FCN and ensemble FCN methods. Figure-6.10(a),(b), and (c) are the ground truth, and prediction results of FCN-8 compressed FCN and ensemble FCN methods for horse rider data-set. Figure-6.10(d),(e) and (f) are the ground truth and predictive results of sheep image, figure-6.10(g), (h), (i), denotes the ground truth and results for high way data-set, figure-6.10(j), (k), (l) are the predictive results for house image. Figure- 6.10(m), (n), (o) are the prediction results of highway image for FCN-ResNet-101, compressed FCN-ResNet-50, and compressed FCN-ResNet-50 skip, as discussed in work referred above. The evaluation matrices and output image have shown that our CNN-EFF method is outperforming for each class object and overall for the selected images in sift-flow and pascal-voc data-sets. In the further section, we have evaluated the effect of CNN and EFF parameters on the accuracy and computational time for the proposed method.

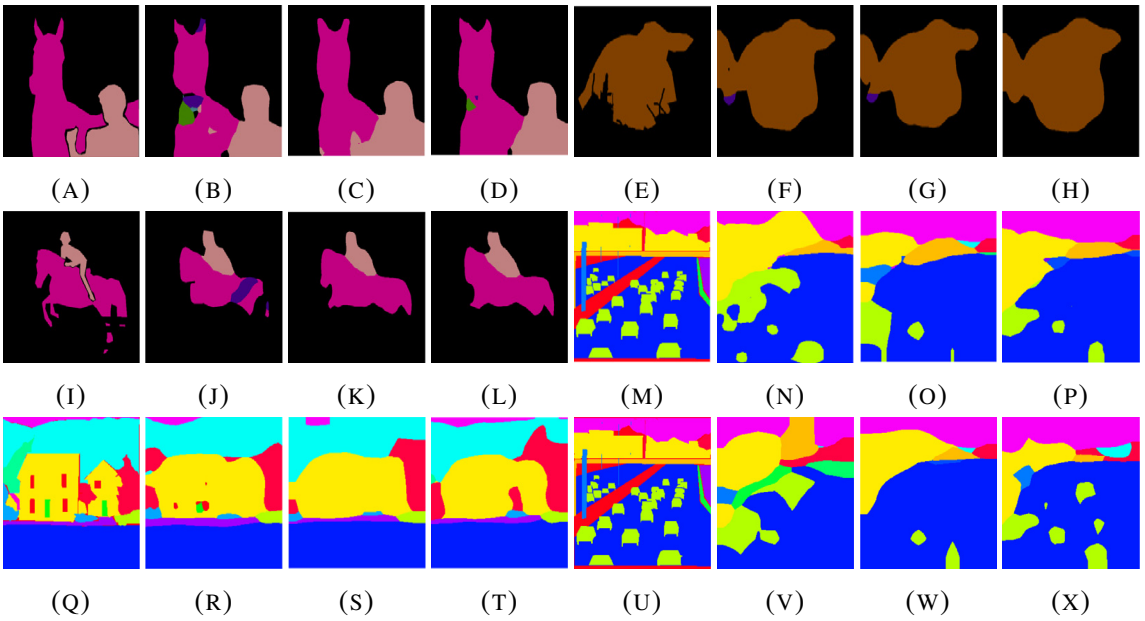


FIGURE 6.10: Results of Comparative approaches in sift-flow, pascal-voc datasets

TABLE 6.3: Comparative Analysis

	Highway	House	sheep	Horse-rider	Horse-keeper
Single-FCN-8	69.6	83.7	70.7	63.6	80.3
Compressed FCN	65.5	83.3	71.0	64.8	80.9
Ensemble	71.2	84.8	77.4	70.8	82.3

6.3.5.2 Conditional random field(CRF) and total variations(TV) based analysis

In this section, we have evaluated the patch-wise CNN(PCNN) with conditional random field(CRF) [158]. CRF is maximum a posterior(MAP) based graph-cut approach which fuses the contiguous spatial information in the labelling process. In most of the state-of-arts, CRF increases significant accuracy by fusing the smoothness cost to data-cost. The data-cost is the probabilistic outcome of our patch-based CNN (PCNN). The approach has denoted as PCNN+CRF. Data cost and smoothness cost have integrated using regularizer $\beta = 2$ in each data-set. The average accuracy(AA) and kappa(K) statistics have shown in table-6.4 for PCNN+CRF. It can be observed in Table-6.1 that the accuracy improvement for the proposed CNN-EFF approach was up to 76.73% and 79.02% for highway and house data sets, but in PCNN+CRF it has increased up to 63.46% only. The table-6.2 has reported the accuracy improvement as 93.43%, 92.31%, and 98.32% accuracy for CNN-EFF, but in PCNN+CRF, the accuracy is 98.45%, 89.74%, and 98.23%. It has found that accuracy has increased for the CNN-EFF model over

TABLE 6.4: $PCNN + CRF$ Results with Smoothness factor β=2 and Accuracies(in %)

Method	Highway		House		Sheep		Horse Rider		Horse Keeper	
	AA	K	AA	K	AA	K	AA	K	AA	K
PCNN+CRF	63.46	0.7964	63.79	0.8282	98.45	1.00	89.74	0.9783	98.23	1.00

TABLE 6.5: $PCNN + TV_{lag}$ method with λ_{Reg}=0.1 and Accuracies(in %)

Method	Highway		House		Sheep		Horse Rider		Horse Keeper	
	AA	K	AA	K	AA	K	AA	K	AA	K
$PCNN + TV_{lag}$	66.72	0.7576	66.68	0.8015	93.54	1.00	90.98	0.9747	98.07	1.00

PCNN+CRF for horse rider and horse keeper data sets. PCNN+CRF method is outperforming the CNN-EFF for sheep data set. The figure-6.11(a) to 6.11(e) denotes the labelling results for the PCNN+CRF method. Subsequently, we have analysed CNN-EFF method with another total variation based method denoted as $PCNN + TV_{lag}$ [180]. In the referred work, the authors have optimized the constrained total variation(TV) based problem. In this section, we have used the class-wise image prediction as a multidimensional input for the total variation based optimization. The objective function for $PCNN + TV_{lag}$ is:

$$\frac{1}{2} norm(x-b) + \lambda \sum_{i,j} norm((x_i - x_j), 1)$$

. Where x is spatial values, and b is the probabilistic values in each class. The Lagrange multiplier(lag) has used to solve the optimization. Table-6.5 has reported the accuracies obtained in the $PCNN + TV_{lag}$ method. The accuracy for highway, house, sheep, hose-rider, and horse-keeper data-set is 66.72%, 66.68%, 93.54, 90.98, 98.07 for $PCNN + TV_{lag}$. CNN-EFF has 10%, 13%, similar, 2%, 0.25% higher inaccuracies than $PCNN + TV_{lag}$ method. Figure-6.11(f) to (j) have shown the classification results of $PCNN + TV_{lag}$

6.3.6 Effect of CNN parameters on Accuracy

In this section, we have compared the different CNN parameters in patch-based training for labelled images obtained from sift-flow and pascal-voc data-sets. Figure-6.12(a) shows that the accuracy is decreasing with an increase in convolution filter size with a step length of one unit. We have kept a deeper filter size fixed and altered the first layer filter size. For the horse-keeper data-set, the CNN method has reported a maximum

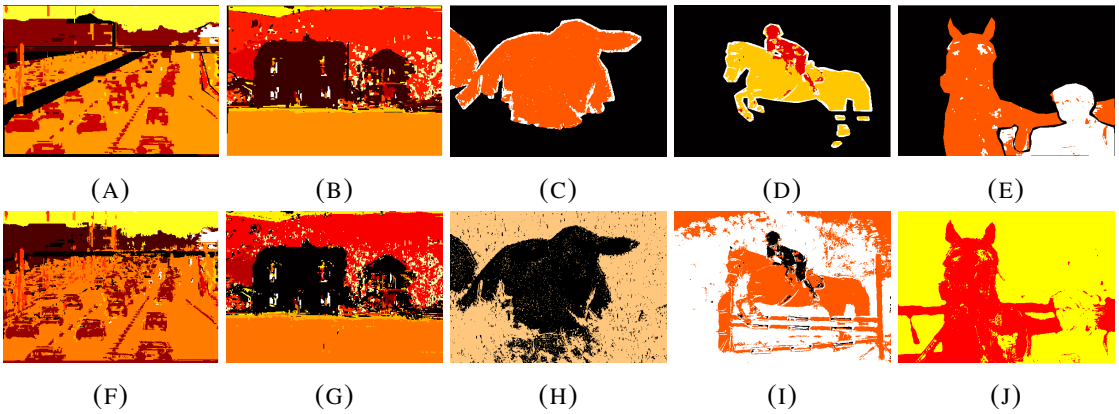
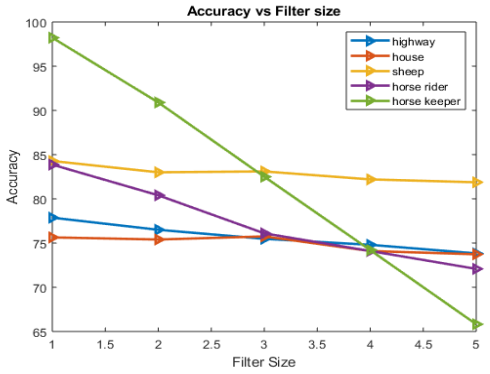


FIGURE 6.11: Results of Comparative approaches for PCNN+CRF from 11-(a) to (e) and for $PCNN + TV_{lag}$ from 11-(f) to (j)

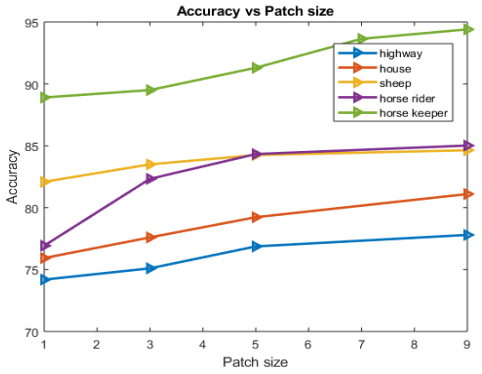
decrease in overall accuracy. In figure-6.12(b), the accuracy is increasing with the increase in the patch-size of the image. We have obtained the accuracy of the square size of 1, 3, 5, 7, and 9-pixel patches. The horse-keeper data-set is achieving a maximum increase in accuracy. An increase in patch-size has increased the computation time of CNN training. In figure-6.12(c), the accuracy is gradually rising on expanding the training sample size of patches. Figure-6.12(d) is showing the accuracy for different images of sift-flow and pascal-voc data-sets for CNN and CNN-EFF methods. The patch size and the training sample size has fixed as nine and 1%, respectively, for accuracy assessment in figure-6.12(d).

6.3.7 Effect of CNN parameters on Computation Time

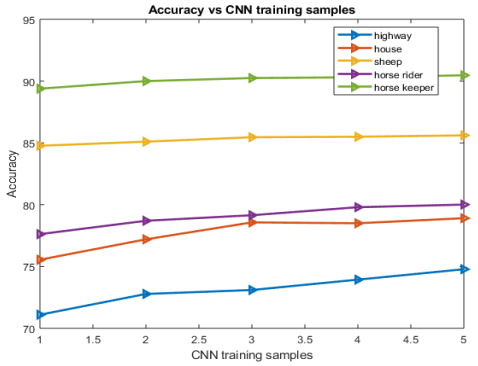
In this section, we have evaluated the impact of CNN parameters on the computation time of training. Figure-6.13(a) has shown the effect of filter size on computation time in which filter size has increased in the same way as the previous section. The computation time is increasing with an increase in the filter size of the first CNN layer. The computation time vs. filter size graph has reported the maximum growth for highway data-set for the CNN method. In figure-6.13(b), the computation time has increased with an increase in patch-size of training samples. The maximum improvement has achieved in the horse keeper data-set. In figure-6.13(c), the computation time is increasing with an increase



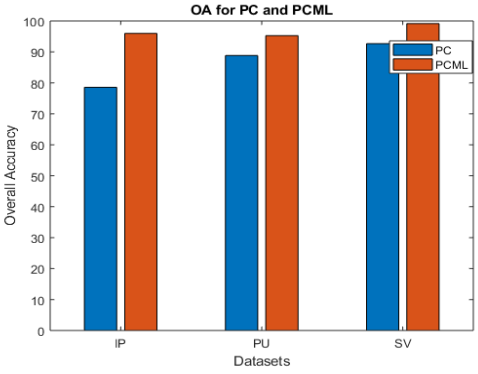
(A) Accuracy vs Kernal Size



(B) Accuracy vs Patch Size



(C) Accuracy vs Training Sample(in %)



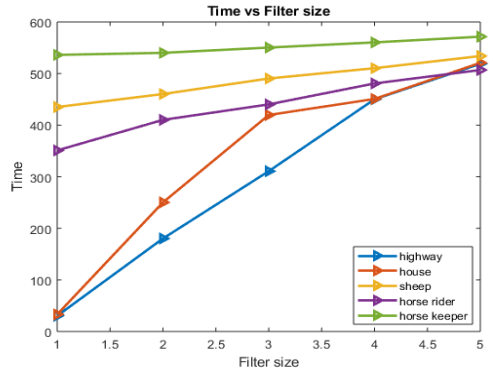
(D) Accuracy for sift-flow and pascal-voc dataset

FIGURE 6.12: Classification Accuracy vs CNN parameters for images of sift-flow and pascal-voc datasets

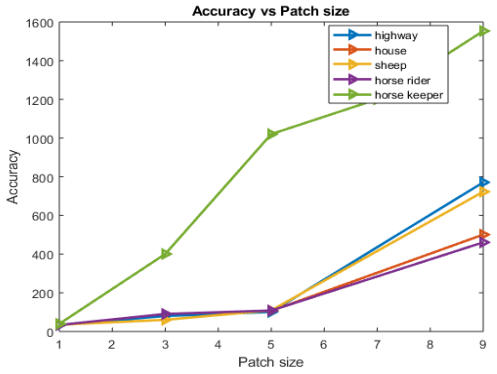
in training sample size. The horse-keeper data-set has reported the maximum increase in computation time. The patch size and training sample size has varied similarly to the previous section. In figure-6.13(d), we have compared the computation time for CNN and CNN-EFF method for each image obtained from sift-flow and pascal-voc data-sets.

6.3.8 Impact of EFF parameters

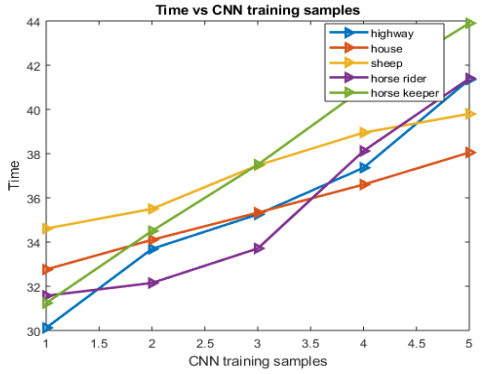
In this section, the parameters of the Jacobian optimization-based EFF method has discussed. The figure-6.14(a) has depicted the number of iterations vs. accuracy plot. It has found that on increasing repetition, there is an increase in the overall accuracy of prediction. We have varied the iterations from one to twenty with a five-step interval for the evaluation



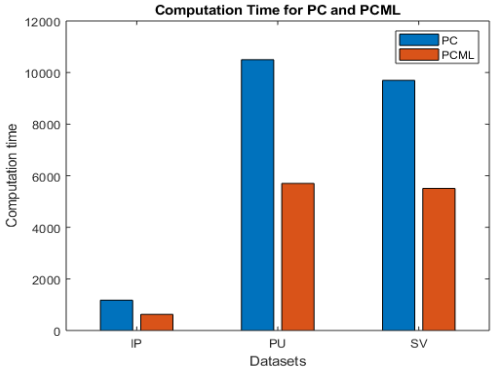
(A) computation time vs Kernal Size



(B) computation time vs Patch Size



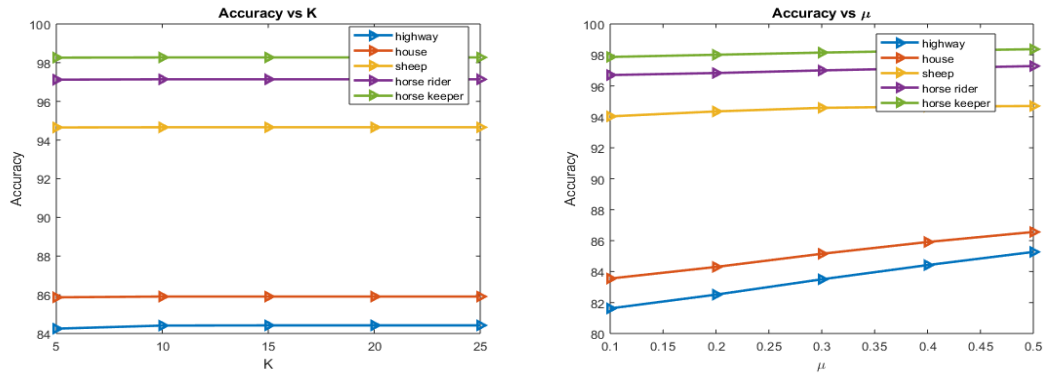
(C) computation time vs Training Sample)



(D) computation time for sift-flow and pascal-voc dataset

FIGURE 6.13: Computation Time vs CNN parameters for images of sift-flow and pascal-voc datasets

of the EFF scheme. Figure-6.14(b) has denoted the accuracy vs. μ, where μ regularizes the weight between the probability and spatial relaxation part of the optimization equation. The accuracy is improving on increasing the value of μ between zero to one. We have reported the more significant improvement in the case of sift-flow images than pascal-voc for EFF based probabilistic improvement. By analysing the parameters of CNN and EFF, we have observed that the CNN and CNN-EFF methods are sensitive to the parameters mentioned above for the accuracy and computation time assessment. Our proposed method, i.e., CNN-EFF, has performed with significant improvement when these parameters have efficiently tuned.



(A) EFF- Accuracy vs iterations (B) EFF- Accuracy vs μ

FIGURE 6.14: Accuracies vs iterations and Accuracy vs μ for sift-flow and pascal-voc datasets

6.3.9 Statistical Test

In this section, we have evaluated the statistical significance in the accuracy of CNN-EFF and CNN methods. McNemar's test is an analytical technique that assesses the difference in two classification models. McNemar is a non-parametric test that performs the t-test on the of-diagonals of the confusion matrix. Let the misclassification rate for model 1 and 2 is r1 and r2, respectively. The hypothesis for the one-sided test of accuracy comparison has denoted as:

$$H_0 : r2 = r1$$

$$H_1 : r2 \neq r1$$

The null hypothesis demonstrates the equivalence of the two models. The one-sided t-test is as follows. n_{11}, n_{22} are diagonal values of confusion matrices. n_{12}, n_{21} are of-diagonal or discordal values. The corresponding t-test is:

$$t = \frac{n_{12} - n_{21}}{\sqrt{n_{12} + n_{21}}} \tag{6.29}$$

if $1 - \phi(t) \leq 0.05$ and ϕ is Gaussian cdf then hypothesis H_0 is rejected. The p values is calculated at x as:

$$t = n_{12}$$

if

$$F_{Bin}(t-1, n_{12}+n_{21}, 0.5) + 0.5 f_{Bin}(t, n_{12}+n_{21}, 0.5) \leq 0.05$$

TABLE 6.6: McNemar's Test with 5% Level of Significance

	h	p	e1	e2
Highway	1	0	0.1911	0.1558
House	1	$2.58 \times 10^{-}279$	0.1708	0.1409
Sheep	1	$7.79 \times 10^{-}160$	0.0624	0.0534
Horse rider	1	$1.08 \times 10^{-}142$	0.0341	0.0286
Horse keeper	1	$2.60 \times 10^{-}175$	0.0226	0.0173

then H_0 has rejected. F_{Bin} and f_{Bin} are the binomial cdf. The classification loss has computed as:

$$e_1 = \sum_{j=1}^{K} \sum_{k=1}^{K} \sum_{i \leq k} \pi_{ijk}$$

$$e_2 = \sum_{i=1}^{K} \sum_{k=1}^{K} \sum_{j \leq k} \pi_{ijk}$$

where $\pi_{ijk} = \frac{n_{(ijk)}}{n_{test}}$ and $\sum_{i,j,k} n_{ijk} = n_{test}$ The outcome of this test is denoted as:

1. h is hypothesis test result that is 0(accept) or 1(reject).
2. p is p-value obtained from one way t-test and lies in (0,1).
3. e1 is classification loss of CNN model.
4. e2 is classification loss of CNN-EFF model.
5. level of significance is 5%.
6. if null hypothesis is true then both models are equivalent and their is no significant increase in accuracy.

The McNemar's test results have shown in Table-6.6. In Table-6.6, the p-value for the Highway data-set is absolute 0, which implies that the null hypothesis was rejected. This signifies that in the CNN-EFF model, the accuracy improvement is practically significant and hence, h=1. If the p-value is near 0.05 and h=0, then both models are equivalent, and there is no improvement in the accuracy. In House, Sheep, Horse rider, and Horse keeper data-sets, the p-values are $2.58 \times 10^{-}279$, $7.79 \times 10^{-}160$, $1.08 \times 10^{-}142$, and $2.60 \times 10^{-}175$ which are close to zero and their h-values are 1. Therefore, in each case, the null hypothesis was rejected, and the test reveals the significant difference in the

accuracy between the CNN and CNN-EFF models. e1, e2 is the classification loss for CNN and CNN-EFF model. The classification losses for Highway and House data-sets are [0.1911,0.1558] and [0.1708,0.1409] which lies between (0,0.2). For sheep, Horse rider, Horse keeper data-sets the classification losses are [0.0624,0.0534], [0.0341,0.0286], and [0.0226,0.0173] which lies in (0,0.07). It has concluded that the significant difference in accuracy has found in each data-set for CNN and CNN-EFF models, but it is maximum for the Highway data-set.

6.4 Conclusions and Future Work

In this work, we have introduced a two-step framework for semantic scene classification and relaxation. In the first stage, an efficient patch-based CNN architecture has investigated, which has provided the prediction probabilities of pixels by a soft-max layer. Subsequently, in step-2, a Jacobian optimization-based spatial label relaxation approach has been applied to improve the prediction probabilities. We have denoted the Jacobian relaxation approach as a post-processing step after CNN testing. The two-step approach has indicated as CNN-EFF, which has evaluated two benchmark semantic image datasets, i.e., sift-flow and pascal-voc. The experimental outcomes have shown that the proposed scheme has dramatically improved the prediction efficiency over the previously proposed methods. The statistical test, accuracy, and computation time-based parameter study and comparative analysis have approved the CNN-EFF scheme as a state-of-art for semantic label relaxation by using deep learning.