Adapting OCR with Limited Supervision

Deepayan Das and C V Jawahar

Centre for Visual Information Technology,
International Institute of Information Technology, Hyderabad, India
deepayan.das@research.iiit.ac.in, jawahar@iiit.ac.in

Abstract. Text recognition systems of today (aka OCRs) are mostly based on supervised learning of deep neural networks. Performance of these is limited by the type of data that is used for training. In the presence of diverse style in the document images (eg. fonts, print, writer, imaging process), creating a large amount of training data is impossible. In this paper, we explore the problem of adapting an existing OCR, already trained for a specific collection to a new collection, with minimal supervision or human effort. We explore three popular strategies for this: (i) Fine Tuning (ii) Self Training (ii) Fine Tuning + Self Training. We discuss details on how these popular approaches in Machine Learning can be adapted to the text recognition problem of our interest. We hope, our empirical observations on two different languages will be of relevance to wider use cases in text recognition.

Keywords: Finetuning · Semi-supervised Learning · Self Training.

1 Introduction and Related Works

At the beginning of the last decade, Deep Learning ushered us into a new era of artificial intelligence. Deep Neural Networks (DNNs) like CNNs [14] and RNNs [10] have shown to learn higher-order abstractions directly from raw data for various machine learning tasks. The need to hand-craft the features for tasks which earlier required domain experts has drastically declined. DNNs have established state-of-the-art results in almost all the computer vision (CV) tasks like image segmentation, object detection, pose estimation etc. Thus, with so much success, it was only natural that DNNs also forayed into one of the oldest computer vision problem of text recognition/OCR.

Motivation One of the crucial steps towards digitizing books is the recognition of the document images using an OCR. More often than not there exists a domain gap between the data on which the OCR was trained and the books on which we want to perform recognition. In the case of digital library creation [1, 2] books that come for digitization will invariably contain variations in their font-style, font-size as well as in their print quality (in case of historical books and manuscripts). Thus an OCR trained on a single source data will invariably fail across such variations. This leads to the inferior performance of the OCR module

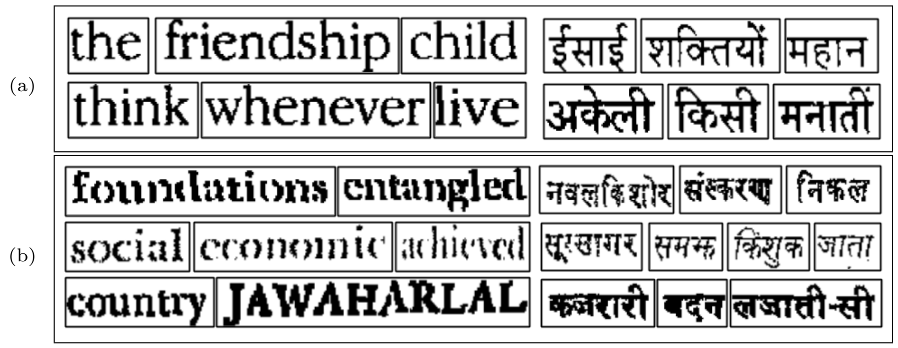


Fig. 2: **(a) Word Images from the collection 1 containing clean-annotated images (b) Word Images from the collection 2 containing partially annotated noisy-degraded images.**

resulting in many wrong words. Thus it becomes essential to fine-tune the OCR model on portions of books that we want to digitize. However, fine-tuning suffers from its inability to learn from unlabelled data as well as an added cost of annotation. Thus, in this paper in addition to fine-tuning, we also explore a branch of the semi-supervised approach called self-training where the OCR can learn from its own predictions, therefore bypassing the need for the creation of annotated data.

Deep learning in text recognition Traditionally, to recognize text from printed documents, sequential classifiers like Hidden Markov Models (HMMs) and graphical models like Conditional Random Fields (CRFs) have been used. These algorithms were popular since they did not require the line images to be explicitly segmented into words and characters, thus reducing the chances of error. However, due to the inability of such algorithms to retain long-term dependencies, Long Short Term Memory networks (LSTMs) emerged as the de facto choice for such sequential tasks. For recognition of handwritten text [9] was one of the earlier works to make use of LSTMs. Later it was adopted by [5, 23, 24] for recognition of printed text for Latin and Indic languages. More recently in [25] CNN was used as a feature extractor in the text recognition pipeline, which helped them achieve the state of the art results on various scene text datasets. Taking the same architecture forward, [12, 6] showed that (CRNN) outperformed Bi-LSTM networks on printed documents. In the last few years attention-based encoder-decoder architecture [17, 15] have also been proposed for character sequence prediction.

Learning with limited supervision Although deep learning algorithms have become very popular, much of its success can be attributed to the availability of large quantities of clean annotated data. Collection of such datasets are both laborious and costly and it proves to be the bottleneck in applying such algorithms. A common approach to mitigate the effects of unavailability of clean and annotated data is to leverage the knowledge gained by DNNs on source tasks for whom labeled examples exist and generalize it to target task which suffers from the unavailability of labeled data. Such approaches fall under the purview of transfer learning which has been widely used in the past especially in Computer Vision (CV) tasks (object detection, classification and segmentation) under a data-constrained setting. Training models from scratch is both resource and time exhaustive. Adapting OCR models by pre-training on ImageNet data and later fine-tuning on Historical documents has been attempted in the past [27]. However, the improvement in the character recognition rate was not very significant. This may be attributed to the difference in image properties between document images and the natural images found in the ImageNet dataset.

Although, fine-tuning improves the performance of DNNs reasonably, yet it suffers from the in consequence of having to annotate a sizeable portion of the data which leads to an added cost. Additionally, fine-tuning also fails to take advantage of the vast amounts of unlabelled data that can be used to enhance the performance of machine learning models. In order to make use of the unannotated data, semi-supervised approaches like pseudo-labeling have become recently popular, where proxy labels are generated for the unannotated images. In [28, 8] the authors showed that by utilizing the unlabeled data in addition to a fraction of the labeled data, they were able to boost the performance of an existing handwriting recognition system (HWR) on a new target dataset. The authors used self-training (discussed in detail in the later sections of this paper) and were able to achieve a performance gain equivalent to a model which was finetuned on the same dataset but with full annotations.
The key contributions of this paper are as follows:

- We study the effect of fine-tuning a pre-trained OCR model on target dataset using a variety of fine-tuning approaches.
- We also, present a self-training approach for adapting OCR to the target data. We show that by combining simple regularization measures like data augmentation and dropout, we can attain improvement in accuracies close to 11% in the case of English and close to 4% in the case of Hindi dataset with no additional manual annotations.
- We also show that by combining the self-training and fine-tuning strategy we can outperform models that have been trained exclusively using the fine-tuning method.
- We empirically support our claims on the dataset for both English and Hindi and show that our proposed approach is language independent.

2 Empirical Verification Framework

Dataset Our dataset is divided into two collections 1) Collection 1: which consists of reasonably clean-annotated images on which our OCR is trained. 2) Collection 2: which consists of partially annotated data containing noisy-degraded images. The images in the collection 1 are significantly different in terms of font style and

Collection	Language	Annotation	Purpose	#Pages	#Lines
Collection 1	English	Yes	Training OCR	1000	14K
	Hindi	Yes	Training OCR	4287	92K
Collection 2	English	Yes	Fine-tuning	50	7K
		Yes	Evaluation	200	9K
		No	Self-training	1100	18K
	Hindi	Yes	Fine-tuning	250	8K
		Yes	Evaluation	1K	20K
		No	Self-training	5K	100K

Table 1: Details of the data used in our work. The table describes the language of the type of collection from which the line images are used. Annotation refers to whether the data split is annotated or not. Also, the column purpose defines the role of each data split.

print quality from the document images in collection 2. This can be observed from Fig 2. The various splits for each collection as well as the purpose of each split is described in Table 1. We annotate a part of collection 2 and split it into two parts (1) fine-tuning (2) evaluation. Our main objective is to transfer knowledge from collection 1 to collection 2 with limited supervision.

Evaluation We use the character recognition rate (CRR) and word recognition rate (WRR) metrics to compare the performance of various models. CRR is the ratio of the sum of edit distance between the predicted text (pt) and ground truth (gt) to the sum of total number characters in pt and gt while WRR is defined as the number of words correctly identified, averaged over the total number of words present in the ground truth.

Implementation Details We use a learning rate of 10^{-5} with a step scheduler. Initially we keep the batch size as 32 and the number of epochs as 100. We use early stopping criterion to avoid overfitting and the optimizer used is Adam.

3 Fine-tuning for text recognition

Image representations learned with CNN on a large scale annotated data can be transferred to other visual tasks that have limited annotated training data [4]. Convolution layers pre-trained on a source task adapt better to the target task as compared to a model trained from scratch. There exist several ways to fine-tune a pre-trained model. The most common approach involves training only the last layer while keeping all the layers frozen. Alternatively, another common approach is to use pre-trained weights of a CNN as initialization, thus fine-tuning all the layers. However, such techniques have gained minimal exposure in the domain of text recognition. In one of the first attempts, [22] reported improvement in model performance when an existing model is fine-tuned on target data as opposed to training from scratch. We study the following fine-tuning approaches.

– **Full**: Using a pre-trained model as a mode of initialization and fine-tune all the layers on the target dataset. We train the model on the annotated portion of collection 2 until the loss on validation data stops decreasing.
– **Last**: Fine-tuning only the recurrent layers while keeping the convolution layers frozen.
– **Chain-thaw**: Fine-tuning one layer at a time while keeping the other layers frozen [7].
– **Unfreeze**: Another variation of chain thaw where we sequentially unfreeze the layers and fine-tune each such instance and finally fine-tune the whole network until convergence [11].

We further assess the effect of learning rate schedulers like slanted triangular learning rate ('Stlr') [11] and cosine annealing scheduler ('Cos') [18] on fine-tuning.

	English		Hindi	
Method	CRR	WRR	CRR	WRR
Base Model	93.65	83.40	91.63	83.95
Full	97.46	95.88	92.90	86.52
Full + Stlr	98.75	98.22	92.99	86.85
Last	96.16	90.88	92.42	85.27
Chain Thaw	97.96	97.53	93.01	86.75
Unfreeze	98.02	97.75	93.03	86.96
Unfreeze + Cos	98.23	98.01	93.11	87.09
Unfreeze + Stlr	**98.79**	**98.46**	**93.20**	**87.40**

Table 2: Character and Word Recognition results for various finetuning methods. The first row shows the CRR and WRR for the pre-trained model. Subsequent rows contain values for models obtained after finetuning the base model.

3.1 Results and Discussions

Table 2 shows the result of various fine-tuning approaches on our English and Hindi datasets. The base model refers to the pre-trained OCR models trained on the clean-annotated source data, which acts as our baseline. Due to the significant difference in the font-style and page image quality between the source and target data, the base model performs poorly as is evident from the results shown. From the experiments, we observe that (1) Fine-tuning the network consistently results in better recognition rates as opposed to training from scratch for both the datasets. (2) Fine-tuning only the recurrent layers ('Last') results in under-fitting, particularly on the English dataset. (3) We observe that fine-tuning the entire network ('Full') seems to give us more favorable results on both the datasets. We believe that this is because the target data contains minute nuances in the font style, which the pre-trained feature extractors are not able to capture well. (4) We also observe that Gradual unfreezing ('Unfreeze') approach to fine-tuning performs the best and is closely followed by 'Chain-Thaw'. 'Chain-thaw' and 'Unfreeze' methods work better than the traditional fine-tuning method since training one layer at a time helps the network to adapt better to the new domain and avoid forgetting. Also, from Figure 4a we observe that validation loss quickly converges for 'Unfreeze' and 'Chain-thaw' techniques with 'Unfreeze' attaining the lowest validation error rate. This confirms the effectiveness of the above two approaches. (5) Additionally, learning rate schedulers consistently boost the performance for 'Full' and 'Unfreeze' methods with 'Unfreeze + Stlr' attaining the best overall accuracies. This is in support of our hypothesis that the network learns better, one layer at a time.

4 Self-training for text recognition

Self-training is one of the widely used approaches in semi-supervised learning. In this method, we generate the prediction for the unannotated data using a pre-trained model and then use them as pseudo-labels to train a new model [19, 21]. Formally, suppose we have m labeled data (L) and n unlabelled data (U) such that $n >> m$. Additionally, L and U do not come from the same source which introduces a domain shift. M_0 is a pre-trained model trained on L. M_0 is used to generate predictions for U such that we now have images along with their predictions $[(I_0, y_0^*), (I_1, y_2^*)....(I_n, y_n^*)]$. The most confident samples are taken and added to the labelled data L. The model M_0 is then trained on the $n + p$ samples and at the end of the training, we obtain model M_1. This process is repeated for a fixed number of cycles such that at the end of each cycle, model M_{i+1} is obtained, which is then used to generate pseudo labels on the remaining $n-p$ unlabelled data samples. The process is continued until there are no more unlabelled samples or when there is no improvement in accuracy on the evaluation dataset. Figure 3 illustrates the self-training approach. We follow the same procedure for training the model using the self-training strategy on our unlabeled dataset.

Although self-training has shown success in a variety of tasks it has the downside of suffering from confirmation bias [16], where the model becomes over-confident on incorrect pseudo labels thus hindering the model's ability to learn and rectify its errors. In [28] the authors proposed to use a lexicon to verify the correctness of the pseudo-labels thereby avoiding the inclusion of any incorrect labels in the training data. However in the absence of a pre-defined vocabulary (in case of highly inflectional languages like Hindi/Tamil) it becomes essential to carefully regularize the network to alleviate the effects of confirmation bias. We use the recently proposed mixup strategy [32] to augment data which is an effective method for network regularization. Additionally, we also provide perturbations to the network using dropout regularization [26], which further

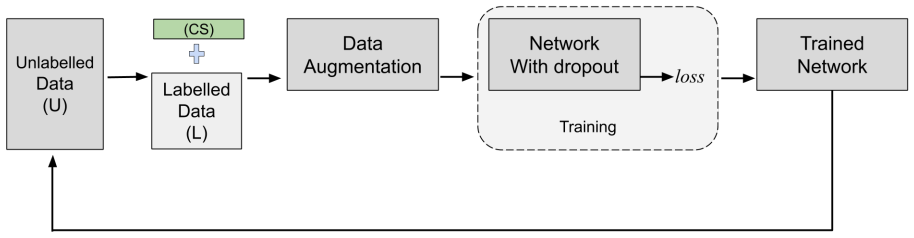


Fig. 3: A pipeline of our proposed iterative self-training approach. At each iteration model, M_i performs inference and confidence estimation on the unannotated dataset. Top k confident samples is mixed with the labelled data L and the combined samples are noised. The network (M_i) is trained on the combined samples at the end of which we obtain model M_{i+1}. The above procedure is repeated until there is no more improvement in word accuracy on test data.

mitigates the confirmation bias and enhances the model's performance. We show that by applying careful regularization measures to our self-training framework, we can achieve comparable accuracies to the fine-tuning approach on the Hindi dataset even though no actual labels were used. We also show that by initially training the network on pseudo-labels followed by fine-tuning on actual labels helped us achieve the best accuracies on both English and Hindi datasets.

4.1 Details

Confidence estimation To estimate the network confidence for each unlabelled sample in U, we take the log probability distribution given by the model M_i over the C classes where each class represents a character present in our vocabulary as shown in equation 1, where p_i is the probability outputted by the RNN at each time step t. We sort the predictions in decreasing order of their confidence scores and take the top 20% at the beginning of each cycle. However, due to the presence of a domain gap between the source and target data, the network tends to confuse between similar-looking characters resulting in a high probability of being assigned to the wrong character. This leads to predictions containing a considerable number of error words. When the network is trained over such confident but wrong predictions, the errors get reinforced in the subsequent trained models. This further amplifies the errors, which shows that the probability distribution can be a poor estimator for selecting the pseudo labels. To neutralize our dependence on the model's probability distribution as confidence estimator, we also take into consideration, the perplexity score of each prediction obtained by a pre-trained language model and finally take a weighted sum over both as shown by equation 2.

$$\text{score} = -\sum_{i=1}^{t} \log(p_i) \tag{1}$$

$$\text{score} = -\alpha \sum_{i=1}^{t} \log(p_i) + (1-\alpha)\frac{1}{m} \log P(w_1, \ldots, w_m) \tag{2}$$

where, $logP(w_i, \ldots, w_t)$ is the joint probability of a sentence and α is the weight parameter that we determine empirically. The language model tends to assign a low score to predicted sentences that have error words, which lead to the overall score being low. This leads to sentences with error words to get weeded out from the confident pseudo labels, which in turn helps the model to avoid confirmation bias while training.

Prediction Ensemble Additionally, while computing the maximum likelihood given by model M_i for each sample, we also take into consideration the probability values outputted by the earlier models i.e. $M_0, \ldots, M_i - 1$ which is given by equation 3 where Z_i is the model prediction at iteration i and z_{j-1} is the average of predictions for models at iteration 0 to $i-1$. Thus, Z contains a weighted average of outputs of an ensemble of models, with recent models having greater weight than the distant models. In the first cycle of our iterative self-training method, both Z and z are zero since no previous models are available. For this reason, we specify λ to be a ramp-up function to be zero on the first iteration. The idea has been borrowed from [13] where it is used to enforce consistency regularization for semi-supervised learning.

$$Z_i = \lambda Z_i + (1-\lambda) z_{i-1} \tag{3}$$

4.2 Regularization

Regularization plays an essential role in our design of the self-training framework. We regularize our network mainly by two ways 1) perturbing the input data and 2) perturbing network. Perturbations to the input data are done by adding Gaussian noise to the input image. In contrast, perturbations to the network are provided by adding a dropout layer, the details of which we discuss in the following sections. Earlier works [30, 3] noted that providing perturbations enforced local smoothness in the decision functions of both labelled and unlabelled data. It has also the added advantage of preventing the model from getting stuck at local minima and avoid overfitting.

Gaussian Noise We multiply the input image with a mask sampled from a binomial distribution. The mask zeros the pixel values at multiple locations, resulting in loss of information. This forces the model to become more robust while making predictions.

Mixup In addition to Gaussian Noise, we also experiment with another type of data augmentation known as mixup [32]. Mixup creates new training samples using a weighted interpolation between two randomly sampled data points $(x_1, y_1), (x_2, y_2)$. where $\lambda \in [0, 1]$ is a random number drawn from a $\beta(\alpha, \alpha)$

Model	Proposed				Baseline			
	English		Hindi		English		Hindi	
	CRR	WRR	CRR	WRR	CRR	WRR	CRR	WRR
VGG + BiLSTMs	**96.53**	**94.01**	**93.04**	**87.23**	93.65	83.40	91.63	83.95
ResNet18 + BiLSTMs	**96.76**	**94.64**	**93.22**	**87.90**	95.23	89.71	92.30	85.97

Table 3: Comparison of word and character recognition rates of CRNN models between baseline and our self training framework.

distribution. Mixup encourages the model towards linear behavior in between training samples. Additionally, mixup has the property of curbing confirmation bias by enforcing label smoothness by combining y_i and y_j as noted by [29]. This is especially important from the perspective of self-training strategy since we are using predictions of unlabelled images as targets. In such cases, the networks, while training, tend to overfit to its predictions.

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j \tag{4}$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j \tag{5}$$

Weight Dropped LSTM In addition to the above data augmentation techniques, we also apply dropout to the recurrent and fully connected network (FCN). The recurrent layers (Bi-LSTMS) is the most fundamental block in any modern OCR models. Therefore, it only makes sense to regularize them for optimal performance. Dropout is one of the most widely used approaches towards regularizing a neural network. However, naively applying dropout to the hidden state affects the RNN's ability to retain long-term dependency [31]. We use Weight Dropped LSTMs proposed in [20] which uses DropConnect on the recurrent hidden to hidden weight matrices. We also introduce dropout on the two fully connected layers after each Bi-LSTM layer with dropout probability set to 0.5. The above data augmentation and model perturbation techniques force the model to act as an ensemble which is known to yield better results than a single network in the ensemble.

We also use slanted triangular learning rates (STLR) proposed in [11] for scheduling our learning rates (LR). The authors argue that STLR enables the network to quickly converge to a suitable region in the parametric space at the start and then gradually refine its parameters.

4.3 Results and Discussions

Table 3 presents the results of our self-training strategy on two kinds of CRNN architectures where the CNN part comes from 1) VGG 2) RESNET18. We observe a significant improvement in word and character accuracies from the baselines for both the models which shows that the refinements that we suggest are model agnostic and can work under any setting. It is interesting to note that our self-training method on the Hindi base model improves the word recognition rate

Heuristics	English		Hindi	
	CRR	WRR	CRR	WRR
ST	94.22	85.12	92.13	85.41
+ STLR	94.72	86.88	92.17	85.45
+ noise	95.61	91.87	92.26	85.57
+ dropout	95.99	92.57	92.26	85.54
+ mixup	**96.48**	**93.57**	**92.57**	**86.23**

Table 4: The breakdown effect of each regularization heuristic on VGG CRNN model

	English		Hindi	
	CRR	WRR	CRR	WRR
ST base	94.22	85.12	92.13	85.41
ST noise	95.61	91.87	92.17	85.47
ST dropout	95.22	89.28	91.91	85.27
ST mixup	96.09	91.73	92.57	86.46

Table 5: Ablation study of various refinements

quite significantly and brings it at par with the 'Unfreeze + Stlr' which is the best performing finetuning approach (shown in Table 2), even though no actual labels were used to train the network. In the case of English, our proposed approach also performs comparatively well with an improvement of 10% in the WRR and 3% in CRR but lags behind the 'Unfreeze' methods. This can be attributed to the difference in the number of training examples between the two datasets with Hindi being far superior in numbers. Thus, we can conclude that the self-training framework benefits from the amount of training data. Additionally, the testing accuracies for each refinement are shown in Table 4. By perturbing input images followed by data augmentation using mixup and adding weighted dropout, we systematically improve the recognition rates at both character and word level. Also, it is essential to note that the values are shown in table 4 are for models that have been trained for only one cycle of our self-training framework and top samples were generated using sum over log probabilities.

Cycles	English		Hindi	
	CRR	WRR	CRR	WRR
Cycle 1	96.49	93.88	92.57	86.23
Cycle 2	96.52	93.92	92.95	87.07
Cycle 3	95.52	93.97	93.04	87.23
Cycle 4	**95.53**	**94.01**	**93.04**	**87.22**

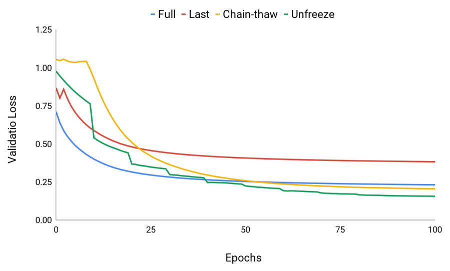
Table 6: Character and word recognition rates at the end of each iterative cycle for both English and Hindi datasets.

scoring	CRR	WRR
prob-dist	96.48	93.45
lang-model	96.01	91.56
weighted score	**96.52**	**93.88**

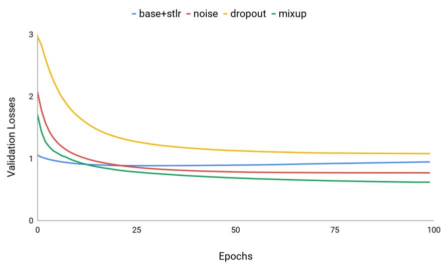
Table 7: Character and word recognition rates for different scoring mechanisms on English dataset

4.4 Observations

To study the impact of individual regularization measures on the performance of our self-trained models we add each regularization one at a time. We then



(a)



(b)

Fig. 5: (a) shows the Validation curves for different fine-tuning methods (b) Validation vs. Epochs plots for the self-training method with various regularizers on English dataset.

train the model with our self-training framework and check the performance for each model on the evaluation dataset. We report the accuracies in Table 5. For the above experiments we use VGG16 architecture and we run on only one cycle of our iterative self-training framework. From our experiments, we observe that (1) only by corrupting the images with Gaussian noise the self-trained attains better recognition rates compared to the base model. The improvement is rather significant especially in the case of English. (2) Similarly, in the case of dropout, WRR and CRR show an improvement of 4% and 2% respectively. However, we observe that in the case of Hindi, the performance drops. We believe that this happened due to the dropout model under-fitting on the Hindi data and believe that a lower dropout probability will easily fix the issue. We also compare the performance self-trained model with and without the *mixup* data augmentation technique. In this study, no other perturbations were provided. We observe that (4) *mixup* improves the character and word recognition rate on both the datasets, Also, from Figure 4b we observe that the validation loss for self-training with mixup strategy converges more quickly than other regularization techniques which demonstrates the effectiveness of the mixup strategy in dealing with the confirmation bias. Additionally, we also study the effect of our iterative self-training strategy and (5) report the gradual increase in accuracy at the end of each self-training cycle in table 6. To show the effectiveness of our proposed weighted sum approach for confidence estimation against log probability distribution and sentence probability score. From Table 7 we observe that the proposed weighted scoring achieves the best accuracies. Since no pre-trained language model was available for the Hindi language, this set of experiments were performed only on the English data. Using only the probability scores from the language model results in an inferior performance. This happens because every language models suffer from an inherent bias towards the dataset on which it was trained. Also, sentences containing proper nouns and abbreviations were given a low score in spite of being correct.

5 Hybrid Approach: Self-training + Fine-tuning

Now, we combine both finetuning and the proposed self-training framework to further improve the performance of our OCR. The success of a finetuning depends on how well weights of the pre-trained models are initialized. In case the source and target tasks are very different or else if the source and target data come from very different distributions, then training the model from scratch seems to be a more viable option as compared to finetuning. In order to acclimatize the weights to the source data, we take advantage of the huge quantities of unlabelled data by employing our self-training strategy. We then follow it by finetuning the self-trained models on the limited annotated data. We show by a series of ablations that this approach outperforms the finetuning approach on both the datasets.

5.1 Results and Discussions

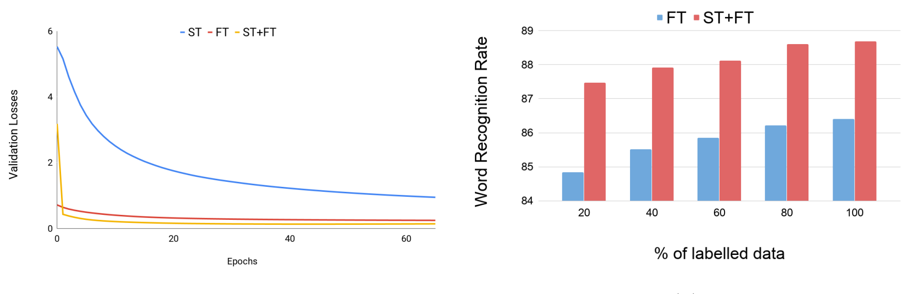


Fig. 7: (a)Validation losses vs Epochs for self-train, fine-tune and hybrid approaches on English dataset. (b)Comparison of word recognition rates between finetuning and hybrid approach when different percentages of labelled data is used

From Table 8 we observe that (1) in the case of Hindi, the improvement in word and character recognition rates are quite significant for our hybrid models with 2% and 1.5% improvement in word accuracy over fine-tuning and self-training methods respectively. (2) In the case of English, the improvement is significant from self-training. However, the increase in accuracies are not noticeable when compared to the fine-tuning approach. We attribute it to the fact that the character and word recognition rates had already maxed out using only fine-tuning, leaving little scope for improvement by the hybrid approach. Fine-tuning on top of self-training has the effect of canceling out any wrong prediction that the model learned during the self-training phase. It also boosts the model's confidence over uncertain predictions. Hence, we always see a spike in character and word accuracy for the hybrid approach. Figure 6a shows the validation curves

Method	English CRR	English WRR	Hindi CRR	Hindi WRR
Fine-tuning	98.76	98.54	92.87	86.41
Self-train	96.53	94.01	93.04	87.23
Hybrid	**98.80**	**98.64**	**93.65**	**88.68**

Table 8: Character and word recognition rates for fine-tuning, self-training and hybrid approach on English and Hindi datasets.

for the above three approaches. We observe that the validation error rate for the hybrid approach reaches minimum within the stipulated number of epochs and it is closely followed by fine-tuning. This is in agreement with the results shown in Table 8 where the hybrid method outperforms the rest of the two methods. Additionally, we also investigate the effect of the amount of data on fine-tuning. For this, we systematically increase the amount of labelled data by a factor of 20% to observe the character and word recognition rates for fine-tuning and hybrid approaches. Figure 6b presents the comparison between the hybrid and the fine-tuning approach for different percentages of labelled data. We observe that by (3) fine-tuning the self-trained model on only a fraction of the labelled data helps us achieve better word recognition rate when compared to the models which were trained using the fine-tuning approach alone.

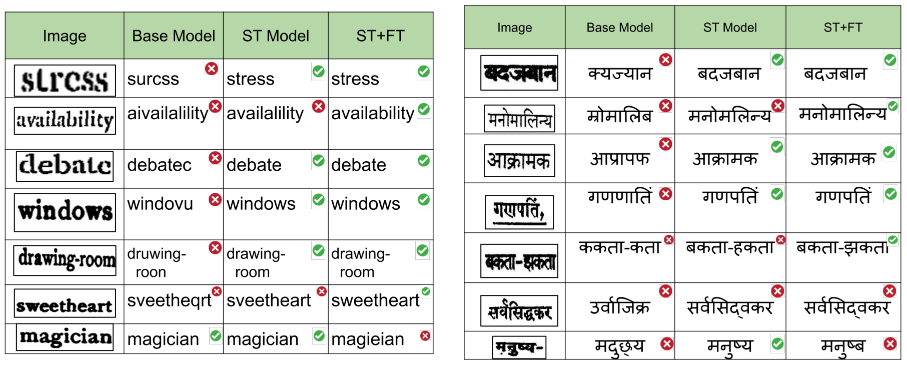


Image	Base Model	ST Model	ST+FT
	surcss ✗	stress ✓	stress ✓
	aivailalility ✗	availalility ✗	availability ✓
	debatec ✗	debate ✓	debate ✓
	windovu ✗	windows ✓	windows ✓
	druwing-roon ✗	drawing-room ✓	drawing-room ✓
	sveetheqrt ✗	sveetheart ✗	sweetheart ✓
	magician ✓	magician ✓	magieian ✗

Image	Base Model	ST Model	ST+FT
	क्यज्यान ✗	बदजबान ✓	बदजबान ✓
	म्रोमालिब ✗	मनोमलिन्य ✗	मनोमालिन्य ✓
	आप्रापफ ✗	आक्रामक ✓	आक्रामक ✓
	गणणातिं ✗	गणपतिं ✓	गणपतिं ✓
	ककता-कता ✗	बकता-हकता ✗	बकता-झकता ✓
	उर्वाजिक्र ✗	सर्वसिद्वकर ✗	सर्वसिद्वकर ✗
	मदुछ्य ✗	मनुष्य ✓	मनुष्ब ✗

Fig. 8: Qualitative Result of our Base, self-trained and hybrid model for English (left) and Hindi (right) datasets. Here ST+FT refers to the model trained using the proposed hybrid approach. We observe that there is a systematic decrease in the character errors from the base model to the hybrid model. This shows that the hybrid model is the best performing model as it has the ability to correct the character errors incurred by the self-trained model due to the confirmation bias. Here crosses show the incorrect words while the correct words are denoted by tick marks.

6 Conclusion

In this paper, we present three different approaches for knowledge transfer between source and target data and compare the performance of each in terms of character and word recognition rates on two different datasets. We also show that simple regularization measure on self-training enables the models to learn more efficiently without getting biased towards the faulty network predictions. Additionally, we also show that combining self-training and fine-tuning can significantly boost the performance of the OCR across datasets of different languages even when the amount of labelled samples is considerably less.

References

1. Google Books. https://books.google.co.in/
2. Project Gutenberg. www.gutenberg.org
3. Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-labeling and confirmation bias in deep semi-supervised learning. ICLR (2019)
4. Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. In: ICMLW. pp. 17–36 (2012)
5. Breuel, T.M., Ul-Hasan, A., Al-Azawi, M.A., Shafait, F.: High-performance ocr for printed english and fraktur using lstm networks (2013)
6. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.: Offline handwriting recognition on devanagari using a new benchmark dataset. In: DAS (2018)
7. Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., Lehmann, S.: Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. ACL (2017)
8. Frinken, V., Fischer, A., Bunke, H., Foornes, A.: Co-training for handwritten word recognition. In: ICDAR (2011)
9. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS (2009)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. (1997)
11. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification (2018)
12. Jain, M., Mathew, M., Jawahar, C.: Unconstrained scene text and video text recognition for arabic script. In: ASAR (2017)
13. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. ICLR (2017)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)
15. Lee, C.Y., Osindero, S.: Recursive recurrent nets with attention modeling for ocr in the wild. In: CVPR (2016)
16. Li, Y., Liu, L., Tan, R.T.: Certainty-driven consistency loss for semi-supervised learning. CoRR (2019)
17. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.: Star-net: a spatial attention residue network for scene text recognition. In: BMVC (2016)
18. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. ICLR (2017)
19. McClosky, D., Charniak, E., Johnson, M.: Effective self-training for parsing. In: ACL (2006)

20. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing lstm language models. ICLR (2017)
21. Reichart, R., Rappoport, A.: Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In: ACL (2007)
22. Reul, C., Wick, C., Springmann, U., Puppe, F.: Transfer learning for ocropus model training on early printed books. CoRR (2017)
23. Sankaran, N., Jawahar, C.: Recognition of printed devanagari text using blstm neural network. In: ICPR (2012)
24. Sankaran, N., Neelappa, A., Jawahar, C.: Devanagari text recognition: A transcription based formulation. In: ICDAR (2013)
25. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. PAMI (2016)
26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR (2014)
27. Studer, L., Alberti, M., Pondenkandath, V., Goktepe, P., Kolonko, T., Fischer, A., Liwicki, M., Ingold, R.: A comprehensive study of imagenet pre-training for historical document image analysis. ICDAR (2019)
28. Stuner, B., Chatelain, C., Paquet, T.: Self-training of blstm with lexicon verification for handwriting recognition. In: ICDAR (2017)
29. Thulasidasan, S., Chennupati, G., Bilmes, J., Bhattacharya, T., Michalak, S.: On mixup training: Improved calibration and predictive uncertainty for deep neural networks. NIPS (2019)
30. Xie, Q., Hovy, E., Luong, M.T., Le, Q.V.: Self-training with noisy student improves imagenet classification (2019)
31. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML (2015)
32. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. ICLR (2017)