Pattern Recognition Letters
journal homepage: www.elsevier.com

Exploring Multi-Tasking Learning in Document Attribute Classification

Tanmoy Mondal[a], Abhijit Das[b], Zuheng Ming[c]

[a]*Mathematical & Electrical Engineering Department, IMT Atlantique, Brest, France*
[b]*Thapar University, Punjab, India*
[c]*L3i, University of La-Rochelle, France*

ABSTRACT

In this work, we adhere to explore a Multi-Tasking learning (MTL) based network to perform document attribute classification such as the font type, font size, font emphasis and scanning resolution classification of a document image. To accomplish these tasks, we operate on either segmented word level or on uniformed size patches randomly cropped out of the document. Furthermore, a hybrid convolution neural network (CNN) architecture "MTL+MI", which is based on the combination of MTL and Multi-Instance (MI) of patch and word is used to accomplish joint learning for the classification of the same document attributes. The contribution of this paper are three fold: firstly, based on segmented word images and patches, we present a MTL based network for the classification of a full document image. Secondly, we propose a MTL and MI (using segmented words and patches) based combined CNN architecture ("MTL+MI") for the classification of same document attributes. Thirdly, based on the multi-tasking classifications of the words and/or patches, we propose an intelligent voting system which is based on the posterior probabilities of each words and/or patches to perform the classification of document's attributes of complete document image.

1. Introduction

Automatic analysis of document attributes such as font is highly useful for several document processing tasks such as character recognition (Simard et al., 2003), document classification, writer and script identification (Shi et al., 2016). Automatic identification of font type, font size and font emphasis can highly improve the accuracy of *Optical Character Recognition (OCR)* systems, especially when the data is processed in multi-script or multi-language form. Contrary to previous techniques in the literature, which have been mainly designed for a single task such as *font type* classification (Tensmeyer et al., 2017), (Cloppet et al., 2018), this work focuses to learn multiple attributes in a multi-tasking environment instead of using a single network for each attribute classification. Given a document image, the motivation of this work is to automatically identify the Font type (e.g *Arial*, *Calibri*, *Courier*, *Times new roman*, *Trebuchet*, *Verdana*), Font size (e.g. 8, 10, 12), Font emphasis (e.g. *bold*, *italic*, *bold-italic*, *none*) and scanning resolution of the document (e.g. 150 dpi, 300 dpi, 600 dpi) image.

To handle the variations of various font characteristics, we compute deep features from the segmented word images and also from the extracted patches. In our first approach, we have applied a MTL system for the classification of document attributes by using either word image or patch images as an input. In our second approach, we have adapted a hybrid CNN of two stream to perform the same multi-tasks by taking segmented word and cropped patch images as the first and second stream of inputs into the network. These two streams act as multi-instances, which are combined by vector-wise operation. We term this proposed MTL and MI based architecture as MTL+MI. The differences among different font types are subtle or even tiny and to capture these differences, we need to operate on local level and that's why we adopted to work on segmented words and/or patches. In addition, we proposed a voting method for the classification of each word and/or patch, which are used as the candidates for voting to decide the final class of the whole document image.

In summary, the contribution of the paper are as follows: i) we present a MTL based framework to classify the font

e-mail: tanmoy.mondal@imt-atlantique.fr (Tanmoy Mondal), abhijit.das@thapar.edu (Abhijit Das), zuheng.ming@univ-lr.fr (Zuheng Ming)

type, font size, font emphasis and scanning resolution of document images. ii) a MTL+MI based framework to jointly learn the tasks with multi-instances i.e. using segmented word and cropped patch images together, iii) we have proposed an intelligent voting system based on the posterior probabilities to perform the classification of the complete document image. The code and dataset used in this research work, can be found in https://github.com/tanmayGIT/Document-Attribute-Classification.git and in http://navidomass.univ-lr.fr/TextCopies/ respectively.

2. Related Works

We review the literature in two direction: the first one is Font type/family and Font size classification and the second one is on multi-tasking network, applied in different domains of computer vision. The existing techniques in the literature about font type/family recognition can be divided into two main family. One is *holistic approach* and other one is *training based approaches*, mainly using deep neural network such as CNN and Recurrent Neural Network (RNN) based techniques. Another technique is proposed for optical font recognition by using typographical features and by using multivariate Bayesian classifier in (Ben Moussa et al., 2010). This approach reported an accuracy of 97.35% over English text lines for 10 font classes. More recently, deep learning techniques based on CNN and RNN have shown very high potential for font classification. By considering font recognition on a single Chinese character is a sequence of classification problem, the authors in (Tao et al., 2016) has proposed principal component based 2D long short-term memory (LSTM) algorithm to classify single Chinese characters into 7 font classes with 97.77% accuracy. Identification of scripts in natural images was proposed in (Shi et al., 2016). The basic idea is combining deep features and mid-level representations into a globally trainable deep model. Another deep CNN based technique are separately used for font type/-family recognition and also for font size recognition in (Amer. et al., 2017). The classification of hand-written Chinese characters into 5 calligraphy classes is performed in (Pengcheng et al., 2017). They have obtained 95% accuracy by using deep features, extracted from a pre-trained CNN on natural images. A competition on the classification of medieval handwriting in Latin script was organized in (Cloppet et al., 2018). The top performing technique obtained an accuracy of 83.9% among 7 methods, submitted in the competition. One recent approach by (Tensmeyer et al., 2017) presents a simple framework based on CNNs. where CNN is trained to classify small patches of text into predefined font classes. Their method achieved a state-of-the art performance on challenging dataset of 40 Arabic computer fonts with 98.8% line level accuracy.

Multi-task Learning (MTL) exploits the task relatedness scenario by learning the common information that is shared between multiple related tasks and promotes sharing of model parameters to exploit the shared information across multiple tasks. The primary issue in MTL setting is to appropriately learn the relation between the tasks (Happy et al.) (Habimana et al., 2020) otherwise can lead to negative performance. Several techniques such as: grouped multiple tasks (Kang et al., 2011), multi-linear relationship networks (Long et al., 2017) has been proposed in the literature. From the literature, it can be concluded that enforcing MTL to a scenario is challenging as it depends on the task behavior and the context of MTL based document analysis has not yet been much explored which encourages us to explore it in detail.

3. Dataset

We have used *L3iTextCopies* dataset (Eskenazi et al., 2015) which has total 42768 document images. For details, see Section. SI in supplementary materials[1]. To obtain the word image boundaries and for cropping these ones, we apply *Google Tesseract* OCR.[2] To avoid the noisy elements, we only have considered the word images of more than 15×15 pixels whereas, the patches are obtained by cropping a window of size (standard input image size of ResNet) 224×224 pixels and by sliding the window by 112×112 in horizontal and vertical directions.

4. Proposed Methods

In this section, we have explained the architectures of our proposed technique for document attributes classification. As the base architecture, we propose a *ResNet50* (He et al., 2016) based model to perform "single task learning" i.e. "Font Emphasis", "Font Type", "Font Size", "Scanning Resolution" tasks separately and independently (see Figure 1a). A pre-trained (trained on *ImageNet* dataset) *ResNet50* model is used here and input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape $(3 \times H \times W)$, where H and W are taken as 224. The images are loaded in a range of $[0, 1]$ and then normalized using $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$. Hence, we obtain a pre-trained feature of size 2048 (see Table. 1 in He et al. (2016)) from *conv5_x* layer.

After obtaining the 2048 features, we add two subsequent layers of : **FC** $(2048 \rightarrow 512)$ (notation like this represents that it is a fully connected layer which has 2048 input nodes and 512 output nodes) and **FC** $(512 \rightarrow 256)$. Then, we add a *batch normalization (BN)* layer which is followed by a layer of: **FC** $(256 \rightarrow 256)$-**RELU**-**BN**, which then get finally connected to one output head (among 4 individual output heads). Each of these heads takes an input of 256 values and provides an output of γ values; where $\gamma = 4$ for *Font Emphasis* task, $\gamma = 6$ for *Font Type* task, $\gamma = 3$ for *Font size* task and $\gamma = 3$ for *Scanning Resolution* task (see Figure 1a). This network is trained and tested by using either segmented word images or the patches.

4.1. Multi Task Learning (MTL)

As the first MTL architecture, we propose a multi task learning network where these four tasks can be performed using a single network (see Fig. 1b). The only difference of this architecture, compared to the previous one is that here last FC layer is finally connected to four individual heads (instead of one head in the previous network, shown in Fig.1a).

[1] throughout this article, "S" stands for "Supplementary Materials"

[2] https://github.com/tesseract-ocr/

4.2. Multi Task and Multi Instance (MI) Learning

As the second MTL based network, we propose to perform a linear combination of the segmented word based MTL network with the segmented patch based MTL network (see Fig. 2). In general, a *MTL+MI* based methods leverage the information which can be learned from other related tasks and it learns a general representation from all the available tasks.

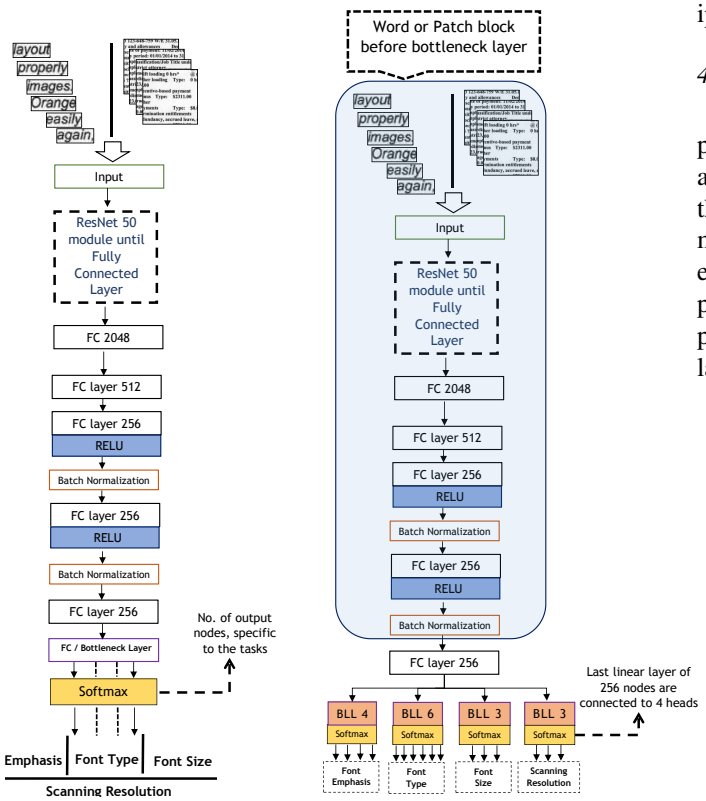


Figure 1. (a) The "Single Tasks Learning" based architecture. (b)The "Multi Tasks Learning" based architecture for document attribute classification.

To combine two *MTL based networks* i.e. *MTL network* for word images and *MTL network* for patches, the proposed architecture (see Fig. 2) remains same until last BN layer of the previous *MTL network* (this block or portion of network is named as *"Word or Patch block before bottleneck layer"* in Fig. 1b). The last **BN** layer has 256 output nodes which are connected to another FC layer of 256 nodes for both the word and patch level networks. Now the block of 256 output nodes (for both the patch and word networks) are copied into 4 heads (i.e. FC layer of word network as well as the patch network are separately connected to 4 heads), where each head is consisting of **FC** $(256 \rightarrow 256)$-**RELU**-**BN**. Then the 1^{st} head of the word network is concatenated to the 1^{st} head of patch network. Same operation is performed for 2^{nd}, 3^{rd} and 4^{th} heads. By each concatenation operation, we concatenate the 256 nodes of word network and 256 nodes of patch network to obtain total 512 nodes. Now each of these 4 group of concatenated 512 nodes are connected to 4 individual and independent head of **FC** $(512 \rightarrow 512)$. Which are then finally connected to 4 individual and independent blocks of **FC** $(512 \rightarrow 256)$-**RELU**-**BN**-**DropOut**. Finally, each head is connected to a **FC** $(256 \rightarrow \gamma)$, followed by **SoftMax** activation, where γ represents the number of output nodes, dedicated to each individual document attributes classifications i.e. for *Font Emphasis*, *Type*, *Size* and *Scanning Resolution* tasks. The objective here is to get benefited from multi instance learning i.e. to train by using both the word and patch images together. Hence, the architecture can get benefited from the equal participation of both the words and patch features together.

4.3. Dynamic Weighed Multi Task & Multi Instance Learning

In the previous architecture (see Section 4.2), we have applied equal weights (or equal participation) to both the words and patch features. But it could also be possible that due to this equal weighted fusion, instead of improving, the accuracy may get decreased. The most probable reason could be the unequal influence of one instance (either words or patches), compared to other. Hence, to avoid this problem, we propose to perform multi instance learning, based on automatically calculated weights, applied to both the instances (see Fig. 3).

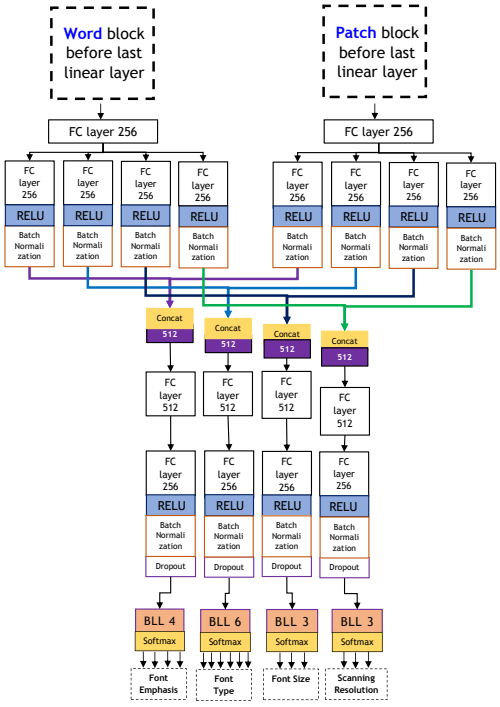


Figure 2. The MTL + MI based architecture

After obtaining 2048 features from word and patch images, we pass them through **FC** $(2048 \rightarrow 512)$. Then each of these 512 features set is passed through several FC layers like : **FC** $(512 \rightarrow 256)$-**RELU**-**BN**; **FC** $(256 \rightarrow 128)$-**RELU**-**BN**; **FC** $(128 \rightarrow 64)$-**RELU**-**BN**; **FC** $(64 \rightarrow 32)$-**RELU**-**BN**; **FC** $(32 \rightarrow 16)$-**RELU**-**BN**; Then 16 output features of word (as well as patch) images are get connected to 4 output heads (see "Word Multi-tasking block" at the left and "Patch Multi-tasking block" at the right of Fig. 3) The weights are automatically computed by initially combining 16 features of "word" network and 16 features of "patch" network together. Then these 32

combined features are individually connected to 4 heads of **FC** $(32 \rightarrow 16)$-**RELU-BN-SoftMax**. Each output head is outputting 2 weight values which are dedicated for the output of "word multi-tasking block" and "patch multi-tasking block" respectively. After obtaining two weight values from each

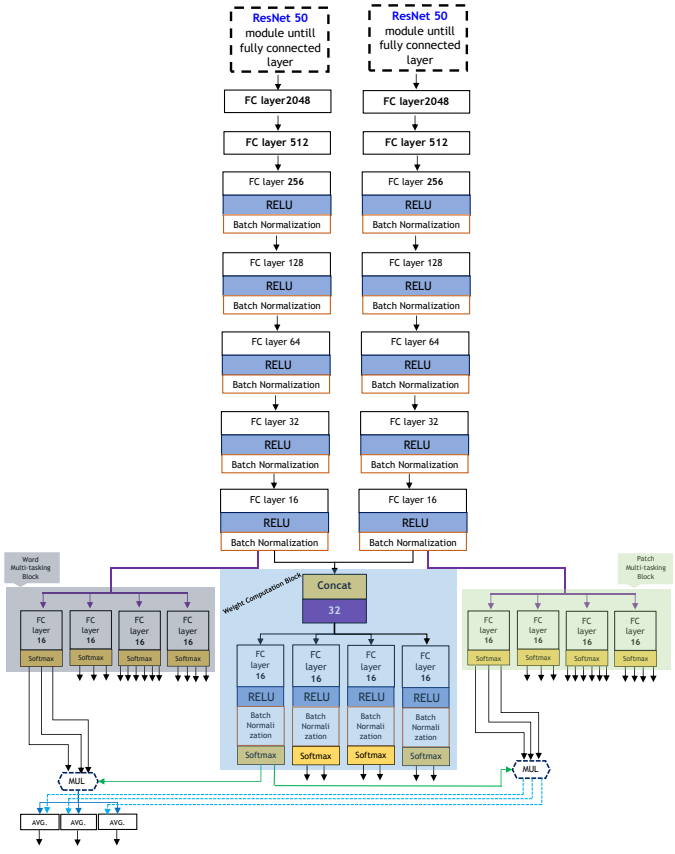


Figure 3. The weighted MTL + MI based architecture

output heads, these ones are multiplied with the outputs of each output heads of "word multi-tasking block" as well as "patch multi-tasking block". For example, the 3 outputs of 1^{st} output head (dedicated for "Font Emphasis" task) of "word multi-tasking block" are get multiplied with the 1^{st} weight value, coming from the 1^{st} head of "weight computation block". In the same manner, the 2^{nd} weight from the 1^{st} head of "weight computation block" got multiplied with the 3 outputs of 1^{st} output head (dedicated for "Font Emphasis" task) of "patch multi-tasking block". Finally, the pair of 3 output values are element-wise averaged to generate 3 final output values, dedicated for "font emphasis" task. In the same manner, we perform the weighted average of other three tasks i.e. *Font Type*, *Font Size* and *Scanning Resolution* tasks.

4.4. Calculation of Loss in Multi Task & Multi Instance Learning

In this section, we will discuss about the loss function of all the above defined 4 different architectures.

a) STL Loss: In the case of *STL* (mentioned in Section 4), the cross entropy loss $\mathcal{L}_{STL}$ is defined as follows :

$$\mathcal{L}_{STL}(\mathbf{X};\Theta) = \sum_{k=1}^{K} -y_k \; logP(y_k|\mathbf{X},\Theta) \tag{1}$$

where K is the total number of output classes, $\mathbf{X}$ and Θ are the input and parameters of the network, y_k is the true label of input $\mathbf{X}$ and $P(y_k|\mathbf{X},\Theta)$ is the predicted probability of class k for the given image $\mathbf{X}$ and parameter Θ. As mentioned in Section 4 that the input could be either segmented word images or patch images whereas the network could be trained for the classification of any of these 4 attributes i.e. for *Font Emphasis*, *Font Type*, *Font Size* and *Scanning Resolution* tasks.

b) MTL Loss: Compared to STL, MTL can gain better performance by jointly learning different tasks. Multi-task learning is an optimization problem for multiple objectives. The loss function of multi-task learning (mentioned in Section 4.1) is calculated by summing up the different tasks in the following manner.

$$\mathcal{L}_{MTL}(\mathbf{X};\Theta) = \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{X};\Theta_t) \tag{2}$$

where T is the number of supervised tasks for which we would like to train in the *MTL* mode, where the loss of each task $\mathcal{L}_t(\mathbf{X};\Theta_t)$ is calculated by using same formula as in Equation 1, X is input of the model and $\Theta = \{\Theta_t\}$ are the parameters of the portion of the network, related to each tasks. In our case, we have 4 independent tasks i.e. *Font Emphasis*, *Font Type*, *Font Size* and *Scanning Resolution* tasks, hence $T = 4$.

c) Multi-Task and Multi-Instance Learning Loss: For the case of *Multi Instance* and *Multi-Task* learning (see Section 4.2), the loss function is calculated in the following manner. Each word image sample, denoted by $\mathbf{X}_i^{word}$ and the patch image sample, denoted by $\mathbf{X}_i^{patch}$, we have T number of label information, where T is the total number of existing tasks. Thus all the tasks, learn together their features in a joint feature space $f \in \mathcal{F}$ through learning the weights for individual tasks. The learned features, before the concatenation operation in Fig. 2 can be represented by $\mathcal{Z}_t^{word_{256}}$, $\mathcal{Z}_t^{patch_{256}}$ as follows:

$$\mathcal{Z}_t^{word_{256}} = \mathcal{F}(\mathbf{X}^{word}, \Theta_t^{word}); \quad \mathcal{Z}_t^{patch_{256}} = \mathcal{F}(\mathbf{X}^{patch}, \Theta_t^{patch}) \tag{3}$$

where Θ_t^{word} and Θ_t^{patch} are the parameters of the word branch and patch branch of each task before the concatenation. Then these feature vectors i.e. $\mathcal{Z}_t^{word_{256}}$, $\mathcal{Z}_t^{patch_{256}}$ are concatenated in the following manner:

$$\mathcal{Z}_t^{concat_{512}} = Concat(\mathcal{Z}_t^{word_{256}}, \mathcal{Z}_t^{patch_{256}}) \tag{4}$$

where $t \in 1,..,T$. After that the *MTL* loss function is calculated in the same manner as :

$$\mathcal{L}_{MTL+MI}^{concat}([\mathbf{X}^{word}, \mathbf{X}^{patch}];\Theta) = \sum_{t=1}^{T} \mathcal{L}_t(\mathcal{Z}_t^{concat_{512}};\Theta_t) \tag{5}$$

where Θ_t is the parameter of the branch of each task after the concatenation of features.

d) Dynamic Weighed MTL+MI Loss: Rather than applying equal weights to both the words and patch features, the dynamic weighted version of *MI + MTL* considers the different contribution of learned words and patch features by weighting the output of word instance $(y_k^{word_t})$ and patch instance $(y_k^{patch_t})$ to formulate the final output value $\mathcal{Y}_{t,k}^{Avg}$ as described in Section 4.3:

$$\mathcal{Y}_{t,k}^{Avg} = Elementwise\_Avg(w_t^1(\Psi) \times y_k^{word_t}, w_t^2(\Psi) \times y_k^{patch_t})$$

where w_t^1, w_t^2 are the weights, associated to the output of each instances i.e. the words ($y_k^{word_t}$) and patch ($y_k^{patch_t}$) respectively. These weights are calculated dynamically during training process i.e. $\{\alpha_i = w_i(\Psi)\}$ are learned automatically by the dynamic weights learning module. Particularly, since the dynamic weights $\{\alpha_i\}$ are the outputs of the softmax layer, where $\sum \alpha_i = 1$ and Ψ ($\Psi \not\subset \Theta$) are the parameters of dynamic weights learning module. The parameters of the dynamic weights learning module (Ψ) and the network parameters (Θ) are optimized simultaneously with the total loss $\mathcal{L}_{MTL+MI}^{weighted}$, given:

$$\mathcal{L}_{MTL+MI}^{weighted} = \sum_{t=1}^{T} \mathcal{L}_{MTL+MI}^{t}([\mathbf{X}^{word}, \mathbf{X}^{patch}]; \Theta) \quad (6)$$

where $\mathcal{L}_{MTL+MI}^{t}$ is the loss of each task with the dynamic weighted output $\mathcal{Y}_{t,k}^{Avg}$:

$$\mathcal{L}_{MTL+MI}^{t}([\mathbf{X}^{word}, \mathbf{X}^{patch}]; \Theta) = \sum_{k=1}^{K} -\mathcal{Y}_{t,k}^{Avg}\, log P(\mathcal{Y}_{t,k}^{Avg}|\mathbf{X}, \Theta) \quad (7)$$

4.5. Component based majority voting for entire document image classification

The training, validation and testing are performed by using either word or the patch images. The classification of an entire document image is done by considering a voting mechanism, where the posterior probabilities of all the word image (or patches) are taken into account to compute the mean posterior probability of each document page. Such a mechanism is shown in Fig. 4, where the word images, obtained from each page are arranged row wise and classes are arranged along columns. The posterior probabilities of each word (or patches) to belong into a specific class is noted in each cells. Hence, by taking column wise mean followed by the maximum of these mean values, we can decide the class of complete document image.

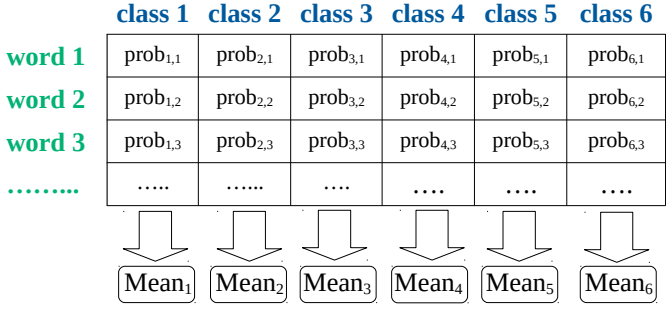


Figure 4. The linear combination of multi-tasking Res-Net of word images and multi-tasking Res-Net of patch images.

Let's say $P(y_k^i|\mathbf{X}_i^p, \Theta)$ is the probability to belong in k^{th} (here in this example, shown in Fig. 4, $k \in 1, ..., 6$) class for the i^{th} word, taken from p^{th} document page of the dataset. The average probability of all the words (belongs to p^{th} document page) for each class is calculated by:

$$\bar{\mathcal{P}} = \frac{1}{n}\left(\sum_{i=1}^{n} P(y_k^i|\mathbf{X}_i^p)\right);\ i \in 1, .., n \quad (8)$$

where there are n number of words exists in p^{th} document page. Now the document page is belongs to which class is decided by computing the maximum of $\bar{\mathcal{P}}$ in the following manner :

$$\mathcal{P}_{max} = max\left(\bar{\mathcal{P}}\right) \quad (9)$$

Thanks to our proposed voting mechanism, even if the predicted posterior probabilities of one or multiple components for the correct class is weaker than the other classes, thanks to the mean based voting scheme, the high posterior probabilities of the remaining components of full page image will compensate and finally the mean accuracy of the true class will be superior.

5. Experiments

In this section, we explain our experimental setting and the results obtained are discussed with detailed analysis. The implementation details of all the above mentioned networks are mentioned in Section 5.2 of supplementary materials.

5.1. Results and discussion

In this section, we have discussed the results of different aforementioned proposed networks. As mentioned in Section 3 that out of 42,768 images, 70% images (29,950) are taken for training and 10% images (4278) are taken for validation. The training data (by considering valid images (see Section 3)) set is consisting of 8,099,561 word images (7,261,574 patches) whereas the validation set is consisting of 1,344,016 word images (1,309,966 patches) respectively. Furthermore, the test set is consisting of 20% of the total images i.e. 8557 images, which gives 5,990,78 word image and 1,324,58 patches.

5.1.1. Accuracies of STL & MTL based networks

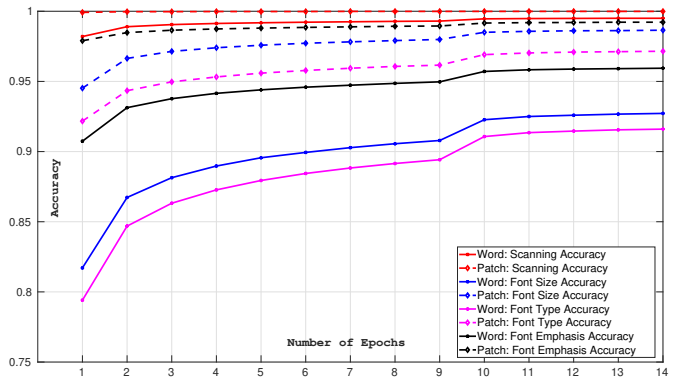
The testing accuracy of STL based network (see Section 4) and MTL based network (see Section 4.1) in shown in Table 1 and the corresponding confusion matrices of these networks are also depicted in Fig.6 and Fig.7 respectively. It can be visible that the patch level accuracy is better than the word level for all the document attribute classifications i.e. for *font type*, *size*, *emphasis* and *scanning resolution*. The *MTL* has performed better than *STL*, which was expected. The training accuracies of MTL based word level as well as patch level networks are shown in Fig. 5 (in the same manner, the training and validation accuracies of MTL based word and patch level networks are shown in Fig. 8 and are discussed in Section SII). It can be visible that the accuracies of all the 4 tasks increases with the iteration of each epochs and patch level network has performed better than word level network.

Table 1. Testing accuracy of STL & MTL based network for word and patch level images

	Top-1 accuracy			
	Font Type	**Font Size**	**Font Emphasis**	**Scanning Resolution**
STL Word	0.8508	0.9095	0.9413	0.9933
STL Patch	**0.9452**	**0.9661**	**0.9780**	**0.9963**
MTL Word	0.8887	0.9137	0.9379	0.9939
MTL Patch	**0.9512**	**0.9760**	**0.9827**	**0.9965**

5.1.2. Details of small dataset formation

To test the accuracy of following two architectures i.e. "Multi-task and Multi-instance Learning" and "Weighted Multi-task and Multi-instance Learning", we choose only a part



(a)

Figure 5. Training accuracies of word based and and patch based MTL networks for "Font type", "Font size", "Font emphasis" and "Scanning resolution" tasks.

Table 2. Training (gray colored row), validation (light blue colored row) and testing (orange colored row) accuracies of MTL and MI based network by combining word and patch level images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Late concat **multiple** FC layers	0.9540	0.9611	0.9766	0.9973
	0.1859	0.4090	0.6757	0.9334
	0.1885	0.4165	0.7217	0.7759
Early concat **multiple** FC layers	0.9680	0.9606	0.9792	0.9975
	0.5287	0.5995	0.6228	0.9740
	0.4918	0.4805	0.6318	0.8147
Early concat **AlexNet** like Conv. layers	0.2929	0.6037	0.5895	0.9754
	0.2221	0.3756	0.5120	0.9438
	0.2197	0.3076	0.4945	0.7998

of the complete dataset in the following manner to obtain faster results and to test various architectural modifications. The document images (also the cropped word and patch images) in the dataset are actually labeled (i.e. the ground truth) in one out of 216 (by combining all the output classes of the four tasks i.e. $4 \times 6 \times 3 \times 3 = 216$) independent classes. We choose 400 word & patch images for training, 100 word & patch images for validation and 150 word & patch images for testing from each of the 216 classes[3]. These limited number of images are chosen by calculating total number of foreground pixels (we have applied adaptive threshold based binarization using Otsu's method) in the image followed by choosing the images which has high number of foreground pixels. Hence in this manner, we choose $83,389$ word and $83,389$ patch images for training and $20,731$ word and $20,731$ patch images for validation.

5.1.3. Accuracy of Multi-task and Multi-instance Learning

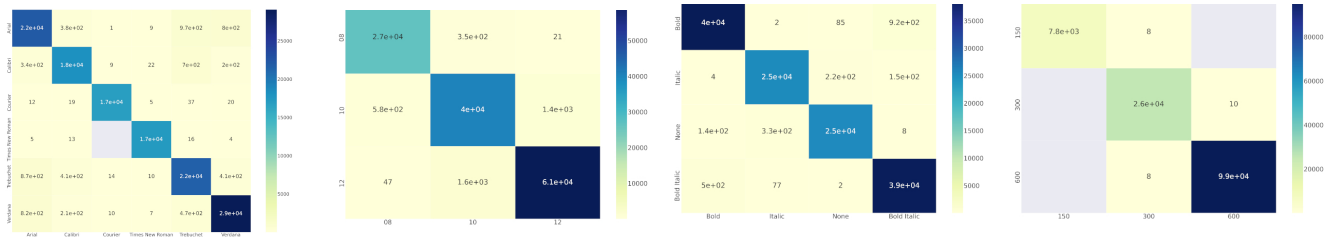
The first test is performed by using the architecture shown in Fig. 2, where the fusion of convoluted features from word images and patch images are performed lately (i.e. the 2048 number of pre-trained "ResNet-50" are passed through several block of FC feed forward networks to finally generate 256 number of features). The results of this network is shown in Table 2 and are mentioned in first row as "Late concat multiple FC layers". Although, we have obtained high training accuracies for all the 4 tasks but the validation and testing accuracies got drastically decreased for all the tasks (whereas the "scanning resolution" task shows comparatively better performance than other three tasks). This is a classical case of "model over-fitting" where the model is able to learn very well on training data but it fails to perform on validation and testing data.

We tried several approaches like increasing and decreasing the percentage of dropout nodes, completely removing all dropout layers, adding more dropout layers at every block of FC layers, removing all the batch normalization layers from the network etc. to overcome this problem. All these trials doesn't helped much and the problem of over fitting sustained. We suspected that the late concatenation of patch and word level features could be the possible reason for this over-fitting problem. Hence, we decided to concat the 2048 number of pre-trained "ResNet-50" features at the early stage of the network. Hence, we designed another network, where 2048 number of pre-trained "ResNet-50" features of word and patch images are concatenated at the early stage to generate 4096 features. The architectural details of this network is mentioned in Section SIV.1 of the supplementary material.

By using this network, although the validation and testing accuracies has highly improved but still there remains a strong gap between the training accuracies (shows unconventionally very high) and testing/validation accuracies of the 4 tasks. By suspecting that the pipeline connection of several FC layers could be the possible reason of this gap between the training and testing/validation accuracies of the 4 tasks, we tried to reduce the number of FC layers. This attempt doesn't helped much and the problem of "model over fitting" still remained. Please look at Table S7 and corresponding discussion in supplementary materials for more details. Furthermore to handle this problem of model over-fitting, we proposed to replace the FC layers by convolutional layers. Here, we adopt the popular "AlexNet" (Krizhevsky et al., 2017) like architecture instead of multiple FC layers, after obtaining 2048 number of pre-trained "ResNet-50" features from word and patch images. We have named this network as "Early concat AlexNet like conv. layers" and it's results are shown in 3^{rd} row of Table 2. The architectural details of this network is mentioned in Section SIV.3.

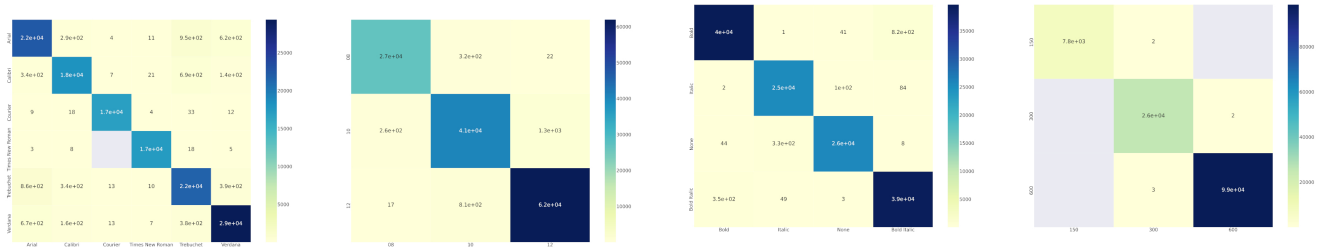
It can be visible from the results, mentioned in Table 2 that although "Early concat AlexNet like conv. layers" network is able to somehow overcome the problem of model over fitting and the difference between training and validation/testing accuracy got decreased (still for "Font Size" task, this difference remains significant), the overall accuracy of the network got significantly reduced. Instead of "AlexNet" like architecture, we have also tried "VggNet" (He et al., 2016) like architecture in the same manner to see whether we can obtain higher accuracies and can also overcome the problem of model "over-fitting". This trial successfully overcome the problem of model "over-fitting" but the overall accuracy of the network also got significantly decreased. Please see Table. S7 and corresponding discussions in supplementary materials for more details.

[3] if there are not enough number of word and/or patch images exists in any of these 216 classes then we first count total number of word (say c_{word}) and patch (say c_{patch}) images in this class and then choose c_{patch} numbers of word and patch images if $c_{patch} < c_{word}$ or vice-versa.



(a) (b) (c) (d)

Figure 6. Confusion matrix of STL patch for: (a) Font Type classification (b) Font Size classification (c) Font Emphasis classification (d) Scanning Resolution classification.



(a) (b) (c) (d)

Figure 7. Confusion matrix of MTL patch for: (a) Font Type classification (b) Font Size classification (c) Font Emphasis classification (d) Scanning Resolution classification.

Table 3. Accuracies of Dynamic weighted MTL and MI based network by combining word and patch level images

	Font Type	**Font Size**	**Font Emphasis**	**Scanning Resolution**
Late concat **multiple** FC layers	0.5265	0.7786	0.8860	0.9866
	0.2976	0.4196	0.4683	0.7484
	0.2898	0.3794	0.4544	0.7000
Late concat **AlexNet** like Conv. layers	0.2854	0.5113	0.5501	0.9427
	0.2114	0.3572	0.3575	0.8673
	0.2038	0.3190	0.3454	0.7679

5.1.4. Accuracy of Dynamic Weighted Multi-task and Multi-instance Learning

The network is shown in Fig. 3 and the corresponding details about this architecture are mentioned in Section 4.3. It can be visible from the accuracies, mentioned at the 1^{st} row in Table 3 that the accuracies are not get further improved, compared to the "Multi-task and Multi-instance Learning" architectures and the problem of "model over-fitting" still remains. Furthermore, to handle the problem of "model over-fitting", we tried to adopt "AlexNet" like architecture by replacing the FC layers with several convolution layers. The details of this network is mentioned in Section SV.1 in the supplementary materials. The results of this network is shown in 2^{nd} row of Table 3. It can be visible from the results that the problem of model over fitting are seems to get resolved for "Font Type" and "Scanning Resolution" tasks but for other two tasks i.e. for "Font Size" and "Font Emphasis" tasks, the model over fitting problem persists. More importantly, the overall accuracies of these 4 tasks become lesser than the "Late concat multiple FC layers" model.

One interesting fact can be noticed from Table. S6 in supplementary materials that even when we are using only word level network (see the 1^{st} row of Table. S6) on the smaller dataset (see Section. 5.1.2), the issue of "model over fitting" persists. This problem can be overcome by using "AlexNet" like layers instead of pipe lined FC layers. The results of this network, named as "Word Level with AlexNet layers" is shown at the 2^{nd} row of Table. S6, where we can see the difference between the training and validation/testing accuracies for these 4 tasks got reduced. But the overall accuracies of all the 4 tasks got decreased than the "Only Word Level" network.

The results of patch level network (denoted as "Patch level with multiple FC layers") on small dataset, the problem of "model over fitting" is not visible (see the 3^{rd} row of Table. S6). Moreover, the accuracies of all the 4 tasks are quite higher compared with it's counter part i.e. "Word level with multiple FC layers" network. We further tested the performance of "AlexNet" like layers instead of pipe lined FC layers and this network is named as "Patch level with AlexNet layers". It can be visible from the results, (see the 4^{th} row of Table. S6) that the problem of "model over fitting" is also disappeared in this case also but the accuracy got decreased, compared to "Patch level with multiple FC layers" network.

Among our various experiments with "MTL + MI" based network, the "Early concat multiple FC layers" network performed better than others. Whereas, the experiments with various weighted "MTL and MI" based networks, the "Late concat multiple FC layers" network performed better. But the accuracies of both these networks i.e. "Early concat multiple FC layers" and "Late concat multiple FC layers" couldn't outperform "Patch Level with multiple FC layers" network. Hence, from all these above mentioned experiments, we can conclude that "Patch Level with multiple FC layers" network is the best performer. The result of this network on the complete dataset is mentioned at the 4^{th} row in Table 1.

The above experiments are performed only on small dataset due to high computational burden which shows poor results, compared to *STL* and *MTL* based networks. To validate our claim, we have experimented *MTL* based networks (named as *"word level with multiple FC layer"* and *"patch level with mul-*

tiple FC layer" in Table. S6) which are also experimented on full dataset (see Table. 1). The results of *MTL* based networks shows superiority over *MTL+MI* and *weighted MTL+MI* based networks, shown in Table. 2 and Table. 3 respectively.

Furthermore, the page level accuracy (evaluated on test dataset only) using the proposed majority voting of "Patch Level with multiple FC layers" network by accumulating the patch level (results shown for word level also) posterior probabilities are mentioned in Table. 4. It can be seen that thanks to our proposed voting scheme, mentioned in Section. 4.5, we have achieved very high page level accuracies in the case of "Patch Level with multiple FC layers" network.

Table 4. Page level testing accuracies of MTL based network using word and patch images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Word Images	0.8879	0.9129	0.9392	0.9935
Patch Images	0.9987	1.00	1.00	1.00

5.2. Implementation details of various proposed networks

In this section, we describe the implementation details of each of the networks which are used for the experiments. We train the proposed network (mentioned in Section 4) by using a pre-trained (trained on *ImageNet* dataset) *ResNet50* model. The model is trained by using stochastic gradient descent with momentum takes as 0.9, multiplicative factor of learning rate decay (denoted by γ) is taken as 0.1, step size is taken as 10, and initial learning rate is taken as 0.0001. We use batches of size of 200 and weight decay 0.0001. The model is trained for 20000 iterations per epoch and the loss function is cross-entropy with a uniform weighing scheme. We have used *GeForce Titan RTX GPUs* with *11GB* of RAM capacity per card. For faster execution, we have used 3 GPU cards in parallel.

5.2.1. Multi task network for segmented words or patches

For this case, almost same network parameters are maintained except the batch size is taken as 800 for word images and 500 for patch images (decided based on the memory capacity per GPU card).

5.2.2. Multi task network for combined segmented words and patches

For this case, almost same network parameters are maintained except the batch size is taken as 100 for both the *multi task and multi instance learning* and as well as for *weighted multi task and multi instance learning*.

In the following Table. 5, we have mentioned the number of trainable and non-trainable parameters of *STL* and *MTL* networks. It can be visible from the Table 5 that we have lot lesser amount of trainable parameters than non-trainable parameters which helps our model to get trained faster. Additionally, it is also clear that MTL has almost similar number of parameters in comparison to STL.

Table 5. Total number of parameters in each network model

Total Parameters	Trainable Parameters	Non-Trainable Parameters
STL		
24,756,035	1,248,003	23,508,032
MTL		
24,759,376	1,251,344	23,508,032

6. Conclusions

In this work, we adhere to explore MTL to perform for 4 document attribute classification tasks i.e. "Font Type", "Font Size", "Font Emphasis", "Scanning Resolution" recognition. In Table. 1, we have shown that "MTL" based networks has outperformed "STL" based network. We further tried to combine the "word and patch" images together by proposing several MTL & MI based network and weighted MTL & MI based networks. But none of them performed better than "MTL" based network, using patch images as the input. Hence, from all the above mentioned experiments, we can conclude that the proposed "MTL" based network, using patch images can attain high accuracy for these 4 classification tasks.

References

Amer., I., Hamdy, S., G., M., 2017. Deep Arabic Font Family and Font Size Recognition. International Journal of Computer Applications 176, 1–6.

Ben Moussa, S., Zahour, A., Benabdelhafid, A., Alimi, A.M., 2010. New features using fractal multi-dimensions for generalized Arabic font recognition. Pattern Recognition Letters 31, 361–371.

Cloppet, F., Eglin, V., Helias-Baron, M., Kieu, C., Vincent, N., Stutzmann, D., 2018. ICDAR2017 Competition on the Classification of Medieval Handwritings in Latin Script. Proceedings of the International Conference on Document Analysis and Recognition, ICDAR 1, 1371–1376.

Eskenazi, S., Gomez-Krämer, P., Ogier, J.M., 2015. When document security brings new challenges to document analysis. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8915, 104–116.

Habimana, O., Li, Y., Li, R., Gu, X., Peng, Y., 2020. A Multi-Task Learning Approach to Improve Sentiment Analysis with Explicit Recommendation. Proceedings of the International Joint Conference on Neural Networks doi:10.1109/IJCNN48605.2020.9206863.

Happy, S., Dantcheva, A., Das, A., Bremond, F., Zeghari, R., Robert, P., . Apathy classification by exploiting task relatedness, in: 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)(FG), pp. 733–738.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. arXiv:1512.03385.

Kang, Z., Grauman, K., Sha, F., 2011. Learning with whom to share in multi-task feature learning., in: ICML, pp. 521–528.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks, in: NIPs, pp. 84–90.

Long, M., Cao, Z., Wang, J., Philip, S.Y., 2017. Learning multiple tasks with multilinear relationship networks, in: Advances in Neural Information Processing Systems, pp. 1594–1603.

Pengcheng, G., Gang, G., Jiangqin, W., Baogang, W., 2017. Chinese calligraphic style representation for recognition. International Journal on Document Analysis and Recognition 20, 59–68.

Shi, B., Bai, X., Yao, C., 2016. Script identification in the wild via discriminative convolutional neural network. Pattern Recognition 52, 448–458.

Simard, P.Y., Steinkraus, D., Platt, J.C., 2003. Best practices for convolutional neural networks applied to visual document analysis. Proceedings of the International Conference on Document Analysis and Recognition, ICDAR 2003-Janua, 958–963.

Tao, D., Lin, X., Jin, L., Li, X., 2016. Principal Component 2-D Long Short-Term Memory for Font Recognition on Single Chinese Characters. IEEE Transactions on Cybernetics 46, 756–765.

Tensmeyer, C., Saunders, D., Martinez, T., 2017. Convolutional Neural Networks for Font Classification. Proceedings of the International Conference on Document Analysis and Recognition, ICDAR 1, 985–990.

Supplementary Materials

SI. Dataset Details

We have used *L3iTextCopies* dataset (Eskenazi et al., 2015) for all the experiments. This dataset is consisting of clean, text-only, typewritten documents which has 22 actual pages. These pages has following characteristics: 1 page of a scientific article with a single column header and a double column body, 3 pages of scientific articles with a double column layout, 2 pages of programming code with a single column layout, 4 pages of a novel with a single column layout, 2 pages of legal texts with a single column layout, 4 pages of invoices with a single column layout, 4 pages of payslips with a single column layout, 2 pages of birth extract with a single column layout. Several variants of these 22 pages are created by combining 6 fonts: *Arial*, *Calibri*, *Courier*, *Times New Roman*, *Trebuchet*, *Verdana*; 3 font sizes: 8, 10 and 12 points; 4 emphasis: *normal*, *bold*, and the combination of *bold* and *italic* which makes the total dataset size of 1584 documents. Then these documents were printed by three printers (*Konica Minolta Bizhub 223*, *Sharp MX M904* and *Sharp MX M850*) then these ones were scanned by three scanners at different resolutions between 150 dpi, 300 dpi and 600 dpi. Which finally generates a complete dataset of total 42768 document images.

SII. Training and Validation accuracies of MTL networks

In the following Fig.8, we have shown the training and validation accuracies of MTL based word level and patch level networks. It can be visible that the accuracies of the 4 tasks increases with the iteration of each epochs.

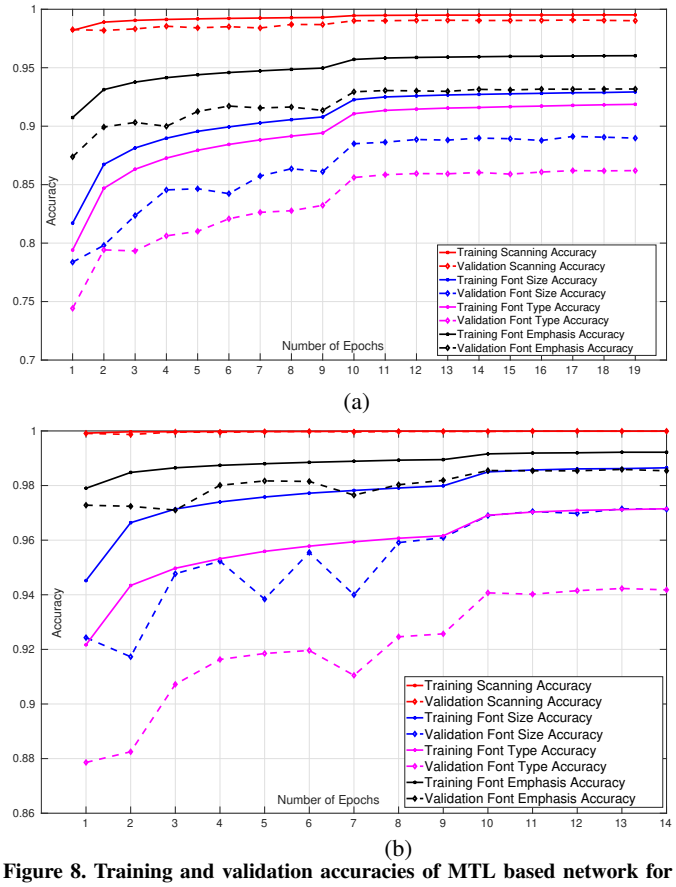


Figure 8. Training and validation accuracies of MTL based network for "Font type", "Font size", "Font emphasis" and "Scanning resolution" tasks : (a) Taking word images as input (b) Taking patch images as input.

SIII. Further experiments on MTL based networks

In continuation with the experiments, mentioned in Table. 1, we further has performed some more experiments on the smaller dataset, mentioned in Section 5.1.2. To get the reference and to validate the results of MTL+MI and weighted MTL+MI based systems on smaller dataset, we needed to test the performance of MTL network on smaller dataset.

SIII.1. Multiple FC layers based MTL network

The first experiment under this category is performed to test the performance of classical MTL network (whose results are shown in Table. 1 and depicted in Fig. 1b) on smaller dataset of cropped word and patch images. The results of this network is shown in Table. S6 and are named as "Word Level with multiple FC layers" and "Patch Level with multiple FC layers" networks.

SIII.2. AlexNet based MTL network

To overcome the problem of model over-fitting and to have the results of reference, here we propose to replace the FC layers of "Multiple FC layers based MTL network" by convolutional layers. We have treated the 2048 pre-trained features as $1D$ channel. Then this $1D$ features are passed through the following 5 blocks of: **CONV** $(in_{ch} \rightarrow out_{ch}; k)$-**BN-RELU**; where the parameters i.e. in_{ch} represents number of input channels, out_{ch} represents number of output channels and k represents the kernel size. Further details of this network is mentioned in Section. SIV.3, where we have proposed another very similar network for MTL+MI based learning. The results of this network is shown in Table. S6.

SIII.3. VggNet based MTL network

In another similar kind of network, we adopt the "VggNet" (He et al., 2016) like architecture instead of "AlexNet" like architecture (the architectural details are mentioned in Section SIV.3). In this network, we have treated the 2048 pre-trained features as $1D$ channel. Then this $1D$ features are passed through the following 2 blocks of:

CONV$(in^1_{ch} \rightarrow out^1_{ch}; k^1)$-**RELU-CONV**
$(in^2_{ch} \rightarrow out^2_{ch}; k^2)$-**RELU-MaxPool**(n)

The parameters are taken as follows in these 2 consequent blocks i.e. for:

1^{st} **block:**
$in^1_{ch} = 2$, $out^1_{ch} = 64$, $k^1 = 3$;
$in^2_{ch} = 64$, $out^2_{ch} = 64$, $k^2 = 3$.

2^{nd} **block:**
$in^1_{ch} = 64$, $out^1_{ch} = 128$, $k^1 = 3$
$in^2_{ch} = 128$, $out^2_{ch} = 128$, $k^2 = 3$.

Table S6. Training (gray colored row), validation (light blue colored row) and testing (orange colored row) accuracies of MTL based network for word and patch level images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Word Level with multiple **FC** layers	0.9703	0.8917	0.9557	0.9828
	0.6073	0.6449	0.7042	0.8431
	0.6986	0.6933	0.8359	0.9386
Word Level with **AlexNet** layers	0.4094	0.4199	0.5860	0.8642
	0.3447	0.4013	0.5203	0.7677
	0.3962	0.3907	0.5992	0.8328
Word Level with **VggNet** layers	0.1762	0.3485	0.2522	0.3459
	0.1740	0.3482	0.2515	0.3479
	0.1751	0.3508	0.2484	0.3507
Patch Level with multiple **FC** layers	0.8631	0.8999	0.9131	0.9933
	0.7751	0.8544	0.9501	0.9979
	0.7777	0.8563	0.9421	0.9899
Patch Level with **AlexNet** layers	0.3688	0.5932	0.6368	0.9444
	0.3802	0.6167	0.7870	0.9806
	0.3600	0.5957	0.8074	0.9562
Patch Level with **VggNet** layers	0.1671	0.3314	0.2519	0.3327
	0.1660	0.3319	0.2488	0.3319
	0.1677	0.3003	0.2446	0.3693

Then the output of this 2^{nd} block is passed through the following 3 blocks of:

$$\textbf{CONV}(in^1_{ch} \rightarrow out^1_{ch}; k^1)\textbf{-RELU-CONV}$$
$$(in^2_{ch} \rightarrow out^2_{ch}; k^2)\textbf{-RELU-CONV}$$
$$(in^3_{ch} \rightarrow out^3_{ch}; k^3)\textbf{-RELU-CONV}$$
$$(in^4_{ch} \rightarrow out^4_{ch}; k^4)\textbf{-RELU-MaxPool}(\mathcal{K})$$

The parameters are taken as follows in these 3 consequent blocks i.e.

for 3^{rd} **block**:
$in^1_{ch} = 128$, $out^1_{ch} = 256$, $k^1 = 3$;
$in^2_{ch} = 256$, $out^2_{ch} = 256$, $k^2 = 3$;
$in^3_{ch} = 256$, $out^3_{ch} = 256$, $k^3 = 3$;
$in^4_{ch} = 256$, $out^4_{ch} = 256$, $k^4 = 3$; $\mathcal{K} = 2$.

For the 4^{th} **block**:
$in^1_{ch} = 256$, $out^1_{ch} = 512$, $k^1 = 3$;
$in^2_{ch} = 512$, $out^2_{ch} = 512$, $k^2 = 3$;
$in^3_{ch} = 512$, $out^3_{ch} = 512$, $k^3 = 3$;
$in^4_{ch} = 512$, $out^4_{ch} = 512$, $k^4 = 3$; $\mathcal{K} = 2$.

For the 5^{th} **block**:
$in^1_{ch} = 512$, $out^1_{ch} = 512$, $k^1 = 3$;
$in^2_{ch} = 512$, $out^2_{ch} = 512$, $k^2 = 3$;
$in^3_{ch} = 512$, $out^3_{ch} = 512$, $k^3 = 3$;
$in^4_{ch} = 512$, $out^4_{ch} = 512$, $k^4 = 3$; $\mathcal{K} = 2$.

The output of the 5^{th} block is a $2D$ features of size 512×64 which is passed through the "Average Pooling" layer[4], having a kernel of size 6. This makes the $2D$ features to get flattened and reduced into the $1D$ feature of size 3072. Then 3072 features are passed through following layers:

$$\textbf{DropOut-FC}((512 \times 6) \rightarrow 768)\textbf{-RELU-DropOut-FC}(768 \rightarrow 384)\textbf{-RELU-FC}(384 \rightarrow 192)$$

[4]here we have used "Adaptive Average Pooling" algorithm from PyTorch library. For more details, see : `https://pytorch.org/cppdocs/api/classtorch_1_1nn_1_1_adaptive_avg_pool1d.html`

After passing through the above mentioned layers, we obtain 192 number of features which are finally connected to 4 output heads, corresponding to 4 individual tasks. We have named this network as "Early concat VggNet like conv. layers" and it's results are shown in Table S6.

SIII.4. Results and Discussion

The results of above described network mentioned in Table. S6, where either word images (named as "Word Level with VggNet layers") or patch images (named as "Patch Level with VggNet layers") are taken as input to the network. It can be visible from these results that the frequently arising model over fitting problem can be clearly resolved in this network but the accuracy of this network for both types of input i.e. either word images or patch images as input, shows inferior accuracies compared to the other networks, shown in Table. S6.

SIV. Further experiments on MTL & MI based networks

In continuation with the experiments, mentioned in Section 5.1.3, where we have tried to replace the late concatenation of patch and word level features by the early concatenation of these features. In the Section 5.1.3, we have discussed such a network where the early concatenated 4096 features are passed through several liner layers, before getting connected to 4 output heads. The detail of this network is mentioned below in Section SIV.1.

SIV.1. Early concat multiple FC layers network

In this network, 2048 number of pre-trained "ResNet-50" features of word and patch images are concatenated to generate 4096 features, which are sequentially passed through several layers like : **FC** $(4096 \rightarrow 2048)$-**RELU-BN**; **FC** $(2048 \rightarrow 1024)$-**RELU-BN**; **FC** $(1024 \rightarrow 512)$-**RELU-BN**; **FC** $(512 \rightarrow 256)$-**RELU-BN**; **FC** $(256 \rightarrow 128)$-**RELU-BN**; **FC** $(128 \rightarrow 64)$-**RELU-BN**; **FC** $(64 \rightarrow 32)$-**RELU-BN**; **FC** $(32 \rightarrow 16)$-**RELU-BN** and finally get connected to 4 output heads. The results of this network are shown at the 2^{nd} row of Table 2 and are named as "Early concat multiple FC layers" network.

Although, by using this network, the validation and testing accuracies got highly improved (see Table. 2) but still there remains a strong gap between the training and validation/testing accuracies. We suspected that may be the pipeline connection of several FC layers could be the possible reason of this gap between training and validation/testing accuracies of 4 tasks, hence we tried to reduce the number of FC layers.

SIV.2. Reduced number of FC layers for MTL & MI based network

Consequently, such a network is explained here. In this network, the pre-trained concatenated 4096 features are sequentially passed through smaller number of FC layers such as: **FC** $(4096 \rightarrow 2048)$-**RELU-BN** which is then directly connected to 4 output heads. This network is named as "Early concat less FC layers_1".

We further modified the network by sequentially passing through the concatenated 4096 features by 2 FC layers as : **FC** $(4096 \rightarrow 2048)$-**RELU**-**BN**; **FC** $(2048 \rightarrow 1024)$-**RELU**-**BN**; which then directly connected to 4 output heads and we named this network as "Early concat less FC layers_2". The results of these two networks are mentioned in 1^{st} and 2^{nd} rows of Table S7 respectively. It can be seen from the validation and testing results of these two networks that the results doesn't get improved for all the 4 tasks and the problem of "model over fitting" still remains.

Table S7. Training, validation and testing accuracies of MTL and MI based network by combining word and patch level images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Early concat **less** FC layers_1	0.9669	0.9642	0.9817	0.9977
	0.4865	0.5204	0.5644	0.9692
	0.5049	0.4258	0.5631	0.8041
Early concat **less** FC layers_2	0.9715	0.9714	0.9836	0.9976
	0.4976	0.4690	0.5661	0.9739
	0.4975	0.4052	0.6156	0.7831
Early concat **VggNet** like Conv. layers	0.1708	0.3457	0.2486	0.3448
	0.1723	0.3451	0.2497	0.3452
	0.1705	0.3683	0.2450	0.3763

SIV.3. AlexNet like convolution layers for MTL & MI based network

To handle the problem of "model over fitting", in Section 5.1.3, we proposed to replace the FC layers by convolutional layers, e.g. adapting "AlexNet" like architecture. We have treated the 2048 pre-trained features of patch and 2048 pre-trained features of word images as two channels of $1D$ vector. Then these two channels are given as input and are passed through the following 5 blocks of: **CONV** $(in_{ch} \rightarrow out_{ch}; k)$-**BN**-**RELU**; where the parameters i.e. in_{ch} represents number of input channels, out_{ch} represents number of output channels and k represents the kernel size. The parameters are taken as follows in these 5 consequent blocks i.e. for:

1^{st} **block:** $in_{ch} = 2$, $out_{ch} = 64$, $k = 11$;
2^{nd} **block:** $in_{ch} = 64$, $out_{ch} = 192$, $k = 5$;
3^{rd} **block:** $in_{ch} = 192$, $out_{ch} = 384$, $k = 3$;
4^{th} **block:** $in_{ch} = 384$, $out_{ch} = 256$, $k = 3$;
5^{th} **block:** $in_{ch} = 256$, $out_{ch} = 256$, $k = 3$;

Finally, the output of 5^{th} block (192 features) are finally get connected to 4 output heads.

SIV.4. VggNet like convolution layers for MTL & MI based network

In another similar kind of network, we adopt the "VggNet" (He et al., 2016) like architecture instead of "AlexNet" like architecture. In this network also, we have treated the 2048 pre-trained features of patch and 2048 pre-trained features of word images as two $1D$ channels. Then these two channels of features are passed through the following 2 blocks of:

CONV$(in^1_{ch} \rightarrow out^1_{ch}; k^1)$-**RELU**-**CONV**
$(in^2_{ch} \rightarrow out^2_{ch}; k^2)$-**RELU**-**MaxPool**(n)

Then the output of this 2^{nd} block is passed through the following 3 blocks of:

CONV$(in^1_{ch} \rightarrow out^1_{ch}; k^1)$-**RELU**-**CONV**
$(in^2_{ch} \rightarrow out^2_{ch}; k^2)$-**RELU**-**CONV**
$(in^3_{ch} \rightarrow out^3_{ch}; k^3)$-**RELU**-**CONV**
$(in^4_{ch} \rightarrow out^4_{ch}; k^4)$-**RELU**-**MaxPool**$(\mathcal{K})$

Table S8. Training, validation and testing accuracies of MTL based network by combining normal patch and noisy patch level images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Early concat **AlexNet** like Conv. layers	0.2995	0.5348	0.5389	0.9667
	0.2883	0.5312	0.6386	0.9914
	0.2592	0.5067	0.6422	0.9691
Early concat **VggNet** like Conv. layers	0.1740	0.3457	0.2521	0.3452
	0.1748	0.3495	0.2524	0.3495
	0.1713	0.3695	0.2435	0.3762

The parameters of these 5 blocks remain same as it is mentioned in Section SIII. In the same fashion, the output of the 5^{th} block is a $2D$ features of 512×64 dimension which is passed through the "Average Pooling" layer, having a kernel of size 6. This makes the $2D$ features to get flattened and reduced into the $1D$ feature of size 3072. Then 3072 features are passed through following FC layers:

DropOut-**FC**$((512 \times 6) \rightarrow 768)$-**RELU**-**DropOut**-**FC**$(768 \rightarrow 384)$-**RELU**-**FC**$(384 \rightarrow 192)$

After passing through the above mentioned layers, we obtain 192 number of features which are finally connected to 4 output heads, corresponding to 4 individual tasks. We have named this network as "Early concat VggNet like conv. layers" and the results are shown in 3^{rd} row of Table S7.

It can be visible from results that the annoying problem of "model over fitting" is also resolved by this network but the overall accuracies of all the 4 tasks are inferior to it's counterpart i.e. "Early concat AlexNet like Conv. layers" network (see Table. 2).

SIV.5. Using noisy patch as input in MTL & MI based network

As it is visible in Table. 2 and are mentioned in Section 5.1.3 that we get highly annoyed by "model over fitting" problem when we tried to combine word and patch images together as the inputs in proposed three networks i.e. "Late concat multiple FC layers", "Early concat multiple FC layers", "Early concat AlexNet like Conv. layers". We have tried several tricks and strategies to overcome the problem of "model over fitting", which are mentioned in Section 5.1.3 and Section SIV.2 and SIV.4. But none of these proposed network architecture could outperform the results of MTL based networks, which uses either word images ("Word Level with multiple FC layers" network) or patch images ("Patch Level with multiple FC layers" network) shown in Table. S6. It can also be seen from Table. S6 that "Patch Level with multiple FC layers" network has shown superior accuracy and has overcome the "model over fitting" problem, compared to "Word Level with multiple FC layers" network. Which inherently imply that word images as the input may be the possible offender for this "model over fitting" problem as well as the reason of decline in accuracy of the results, shown in Table. 2 and Table.S7.

Hence, in the following setup, we have replaced cropped word images by noisy patch images (get inspired from "denoising auto encoder" [5]). Hence, we tested the performance of "Early concat AlexNet like Conv. layers" network (described in 2^{nd} last paragraph in Section 5.1.3) and "Early concat VggNet like Conv. layers" network (described in Section SIV.4) by considering patch images as the 1^{st} input and noisy version of the same patch images (we have applied standard Gaussian noise[6] of mean=0 and standard deviation = $\sqrt{0}.1$) as the 2^{nd} input of the network. The data is partially corrupted by noises and are added to the input vector in a stochastic manner. Then the network is trained to correctly classify even if it is trained with noisy images.

The results of these networks are shown in Table.S8. It can be visible from the results that the problem of "model over fitting" is resolved here also. Moreover, the results of "Early concat AlexNet like Conv. layers" network got highly improved in comparison with it's counterpart in Table. 2. Whereas, the results of "Early concat VggNet like Conv. layers" network from Table.S8 remains same as it's counter part, mentioned in Table. S7. Most probable reason could be the inherent architecture of "VggNet", which is able to overcome the "model over fitting" problem, irrespective of word images (which is the most probable offender for "model over fitting" problem) as input to the network. But in particular, both of these networks couldn't outperform the "Patch Level with multiple FC layers" network, mentioend in Table. S6.

Table S9. Training, validation and testing accuracies of weighted MTL based network by combining normal patch and noisy patch level images

	Font Type	Font Size	Font Emphasis	Scanning Resolution
Late concat **multiple** FC layers	0.5025	0.7604	0.8432	0.9905
	0.4469	0.6948	0.8364	0.9884
	0.3017	0.4366	0.5966	0.7374
Late concat **AlexNet** like Conv. layers	0.2946	0.5657	0.5791	0.9823
	0.2614	0.5224	0.6412	0.9892
	0.2012	0.4742	0.4517	0.9302

SV. Further experiments on weighted MTL & MI based learning

In the following section, we have explained further details about several experiments regarding weighted MTL & MI based learning.

SV.1. AlexNet like Weighted Multi-task and Multi-instance Learning

Here we have mentioned the details about "Late concat AlexNet like Conv. layers" network, whose results are given in Table. 3. After obtaining 2048 pretrained "ResNet-50" features from word and patch images, we pass them through a FC layer of: **FC** $(2048 \rightarrow 512)$ to reduce the dimension of feature. After obtaining 512 features of word image network and 512 features of patch image network, we treat each of them as a channel of $1D$ vector. Then each of these channel are passed through the 5 blocks of: **CONV** $(in_{ch} \rightarrow out_{ch}; k)$-**BN**-**RELU**; where the parameters i.e. in_{ch} represents number of input channels, out_{ch} represents number of output channels and k represents the kernel size. The parameters are taken as before like "Late concat AlexNet like conv. layers" network, mentioned in Section 5.1.3.

After passing through the 5^{th} block, we obtain the $2D$ features of 256×124 dimension. This $2D$ feature is passed through the "Average Pooling" layer (having kernel of size 6), which scale down the feature into of dimension 256×6. Then this reduced dimensional feature is flatten[7] to get $1D$ features of size 1536 which is then passed through 3 blocks of following layers: **DropOut**$(\mathcal{P} = 50)$-**FC** $(1536 \rightarrow 768)$-**RELU**; **DropOut**$(\mathcal{P} = 50)$-**FC** $(768 \rightarrow 384)$-**RELU**; **FC** $(384 \rightarrow 192)$; After passing through these 3 blocks, we finally obtain 192 features from the word network (using word images only) as well as from the patch network (using patch image only). The 192 output features of word network get connected to 4 output heads (for reference, see "Word Multi-tasking block" at the left of Fig. 3). Another set of 192 features from patch images are also get connected to 4 output heads (for reference, see "Patch Multi-tasking block" at the right of Fig. 3). The weights are automatically computed in the same manner by initially combining 192 features of "word" network and 192 features of "patch" network together. Then these 384 number of combined features are individually connected to 4 heads of: **FC** $(384 \rightarrow 192)$-**RELU**; **FC** $(192 \rightarrow 96)$-**RELU**; **FC** $(96 \rightarrow 48)$-**RELU**; **FC** $(48 \rightarrow 24)$-**RELU**; **FC** $(24 \rightarrow 2)$-**RELU** (take reference from the Fig. 3 where you can see that each output head is outputting 2 weight values). Hence, from each output head, we can get two weight values where the 1^{st} weight value is dedicated for the output of "word multi-tasking block" and the 2^{nd} weight value is dedicated for "patch multi-tasking block". After obtaining two weight values from each output head, dedicated to *Font Emphasis*, *Font Type*, *Font Size* and *Scanning Resolution* tasks, these ones are multiplied and averaged with the outputs of each output heads of "word multi-tasking block" as well as "patch multi-tasking block" in the same manner as it is shown and described in Fig. 3. The results of this network is mentioned in Table. 3 and the corresponding description is given in Section 5.1.4.

In the above Table. S10, we have also experimented the effect of "soft-max" layer while computing the weight values from each output heads. Which means, we have removed the "soft-max" layers in both the network i.e. "Late concat multiple FC layers" and "Late concat AlexNet like Conv. layers" (see Table. 3) networks from all 4 output heads of "Word Multi-tasking Block" and "Patch Multi-tasking Block", shown in Fig.3. It can

[5] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning (2016), The MIT Press

[6] `https://gist.github.com/Prasad9/28f6a2df8e8d463c6ddd040f4f6a028a#gistcomment-2857098`

[7] we have used "nn.Flatten()" function of PyTorch library to flatten the feature. For more details, please see : `https://pytorch.org/docs/stable/generated/torch.nn.Flatten.html`

Table S10. Training, validation and testing accuracies of weighted MTL based network by combining word and patch level images (without soft-max layer for weight calculation)

	Font Type	**Font Size**	**Font Emphasis**	**Scanning Resolution**
Late concat **multiple** FC layers	0.4887	0.6970	0.6511	0.9224
	0.2558	0.4053	0.4091	0.7032
	0.2417	0.3733	0.3789	0.5368
Late concat **AlexNet** like Conv. layers	0.1824	0.3823	0.2631	0.6090
	0.1790	0.3701	0.2476	0.5713
	0.1828	0.3517	0.2689	0.5984

be clearly visible from the results in Table. S10 in comparison with the results, shown in Table. 3 that when used during the weight computation, the “soft-max” layers plays a vital role in improving accuracy.