Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation

Sijin Li
sijin.li@my.cityu.edu.hk

Weichen Zhang
wczhang4-c@my.cityu.edu.hk

Antoni B. Chan
abchan@cityu.edu.hk

Department of Computer Science
City University of Hong Kong

Abstract

This paper focuses on structured-output learning using deep neural networks for 3D human pose estimation from monocular images. Our network takes an image and 3D pose as inputs and outputs a score value, which is high when the image-pose pair matches and low otherwise. The network structure consists of a convolutional neural network for image feature extraction, followed by two sub-networks for transforming the image features and pose into a joint embedding. The score function is then the dot-product between the image and pose embeddings. The image-pose embedding and score function are jointly trained using a maximum-margin cost function. Our proposed framework can be interpreted as a special form of structured support vector machines where the joint feature space is discriminatively learned using deep neural networks. We test our framework on the Human3.6m dataset and obtain state-of-the-art results compared to other recent methods. Finally, we present visualizations of the image-pose embedding space, demonstrating the network has learned a high-level embedding of body-orientation and pose-configuration.

1. Introduction

Human pose estimation from images has been studies for decades. Due to the dependencies among joint points, it can be considered a structured-output task. In general, human pose estimation approaches can be divided by two types: 1) prediction-based methods; 2) optimization-based methods. The first type of approach views pose estimation as a regression or detection problem [18, 31, 19, 30, 14]. The goal is to learn the mapping from the input space (image features) to the target space (2D or 3D joint points), or to learn classifiers to detect specific body parts in the image. This type of method is straightforward and usually fast in the evaluation stage. Toshev *et al*. [31] trained a cascaded network to refine the 2D joint locations in an image stage by stage. However, this approach does not explicitly consider the structured constraints of human pose. Followup work [14, 30] learned the pairwise relationship between 2D joint positions, and incorporated them into the joint predictions. Limitations of prediction-based methods include: the manually-designed constraints might not be able to fully capture the dependencies among the body joints; poor scalability to 3D joint estimation when the search space needs to be discretized; prediction of only a single pose when multiple poses might be valid due to partial self-occlusion.

Instead of estimating the target directly, the second type of approach learns a score function, which takes both an image and a pose as inputs, and produces a high score for correct image-pose pairs and low scores for unmatched image-pose pairs. Given an input image x, the estimated pose y^* is the pose that maximizes the score function, i.e.,

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f(x, y), \tag{1}$$

where $\mathcal{Y}$ is the pose space. If the score function can be properly normalized, then it can be interpreted as a probability distribution, either a conditional distribution of poses given the image, or a joint distribution over both images and joints. One popular model is pictorial structures [9], where the dependencies between joints are represented by edges in a probabilistic graphical model [16]. As an alternative to generative models, structured-output SVM [32] is a discriminative method for learning a score function, which ensures a large margin between the score values for correct input pairs and for incorrect input pairs [24, 10].

As the score function takes both image and pose as input, there are several ways to fuse the image and pose information together. For example, the features can be extracted jointly according to the image and poses, e.g., the image features extracted around the input joint positions could be viewed as the joint feature representation of image and pose [9, 26, 34, 8]. Alternatively, features from the image and pose can be extracted separately and concatenated, and the score function trained to fuse them together [11, 12]. However, with these methods, the features are hand-crafted, and performance depends largely on the quality of the features.

On the other hand, deep neural networks have been shown to be good at extracting informative high-level fea-

tures [27, 3]. In this paper, we propose a unified framework for maximum-margin structured learning with deep neural network for human pose estimation. Our unified framework jointly learns the image and pose feature representations and the score function. In particular, our network first extracts separate feature embeddings from the image input and from the 3D pose input. The score function is then the dot-product between the image and pose embeddings. The score function and feature embeddings are trained using a maximum-margin criteria, resulting in a discriminative joint-embedding of image and 3D pose. The dot-product score function is efficient to compute, and allows for fast inference over a large set of candidate poses. In addition, our proposed framework is quite general and can be applied to a wide range of structured-output tasks.

2. Related work

Here we review recent related works in deep neural network and structured learning.

2.1. 2D pose estimation via detection with deep networks

Traditional pictorial structure models usually apply linear filters on hand-crafted features, e.g., HoG and SIFT, to calculate the probability of the presence of body parts or adjacent body-joint pairs. As shown in [8], the quality of the features are critical to the performance, and, while successful for other tasks, these hand-crafted features may not be necessarily optimal for pose estimation. Alternatively, with sufficient data, it is possible to learn the features directly from training data. In recent years, deep neural networks, especially convolutional neural networks (CNN), have been shown to be effective in learning rich features [23, 17]. Jain *et al*. [14] trains a CNN as a sliding-window detector for each body part, and the resulting body-joint detection maps are smoothed using a learned pairwise relationship between joints. Tompson *et al*. [30] extends [14] by feeding the body-joint detection maps into a modified convolutional layer that performs pairwise smoothing, allowing feature extraction and pairwise relationships to be jointly optimized. Chen *et al*. [5] uses a deep CNN to predict the presence of joints and the pairwise relationships between joints, and the CNN output is then used as the input into a pictorial structure model for 2D pose estimation.

The advantage of these approaches is that the features extracted by deep networks usually lead to better performance. However the detection-based methods for 2D pose estimation are not directly applicable to 3d pose estimation due to the need to discretize a large pose space – the number of joint positions grows cubicly with the resolution of the discretization, making inference computationally expensive [4]. In addition, it is difficult to predict 3D coordinates from only a local window around a joint, without any other contextual information.

2.2. Pose regression via deep networks

In contrast to detection-based methods, regression-based methods aim to directly predict the coordinates of the body-joints in the image. Toshev *et al*. [31] trains a cascade CNN to predict the 2D coordinates of joints in the image, where the CNN inputs are the image patches centered at the coordinates predicted from the previous stage. Li *et al*. [19] use a multi-task framework to train a CNN to directly predict a 2D human pose, where auxiliary tasks consisting of body-part detection guide the feature learning. This work was later extended for 3D pose estimation from single 2D images [18].

One disadvantage of regression-based methods is that they can only predict one pose for a given image. This may cause difficulties on images where the pose is ambiguous due to partial-self occlusion, and hence several poses might be valid. In contrast, our proposed model is better able to handle ambiguities since several valid image-pose pairs can have similar high scores.

2.3. Structured-output prediction and feature embedding

Rodríguez [24] represents the score function between word labels and images as the dot-product between the word-label feature and an image embedding, and trains a structured SVM (SSVM) to learn the weights to map the bag-of-words image features to the image embedding. Dhungel *et al*. [7] uses structured learning and deep networks to segment mammograms. First, a network is trained to generate a unary potential function. Next, a linear SSVM score function is trained on the output of the deep network, as well as other potential functions. Osadchy *et al*. [22] apply structured learning and CNN for face detection and face pose estimation. The CNN was trained to map the face image to a manually-designed face pose space. A per-sample cost function is defined with only one global minimum so that the ground-truth pose has minimum energy. In contrast to [7, 24, 22], we learn the feature embedding and score prediction jointly within a maximum-margin framework.

Jaderberg *et al*. [13] proposed a deep structured-output network for recognizing text in images. The score function is a conditional random field (CRF), where the input is an image and the output is a word. The unary and higher-order potential functions of the CRF are two CNNs, which are trained to recognize single characters and n-grams in the image, and the framework is jointly trained with a maximum margin cost. In the context of pose recognition, [13] is a pictorial structure model with higher-order terms, whereas our method is similar to learning a non-linear embedding with a linear SSVM score function. In particular, the main difference is that we do not manually-design the score function to encode the output structure as pairwise or higher-order

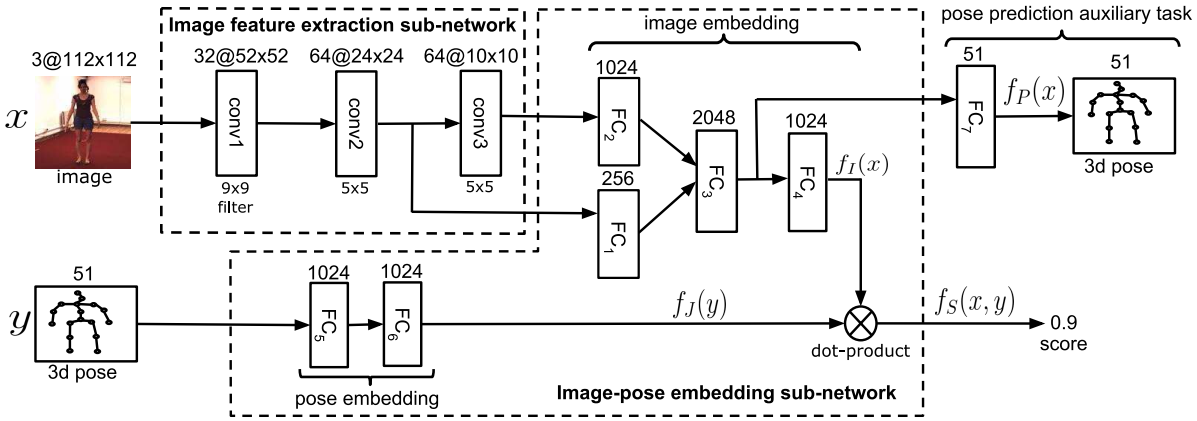

Figure 1. Deep-network score function. The image input is fed through a set of convolutional layers for image feature extraction. Two separate sub-networks are used to embed the image and the pose into a common space, and the score function is the dot-product between the two embeddings. An auxiliary 3D body-joint prediction task is used to guide the network to find good image features. Each convolutional layer is followed by a max-pooling layer, which is not drawn to reduce clutter.

terms (i.e., the CRF), but instead train the network to learn both image and pose embeddings such that a score function can be represented as dot-product. Furthermore, the internal image representations in [13] are strongly supervised, consisting of character/n-gram classifiers, whereas the internal representations (image/pose embeddings) in our method are learned from the data. Although both methods use a maximum-margin cost, [13] uses a fixed margin for all input/output pairs, whereas our method uses margin rescaling.

2.3.1 Unsupervised joint feature embedding

Deep networks have also been used to learn joint embeddings for multi-modal inputs. Ngiam *et al*. [21] embed audio-video pairs by jointly training autoencoders with a shared middle layer. Pereira *et al*. [28] build a generative model for image-text pairs by adding a binary hidden layer on top of image-specific and text-specific deep Boltzmann machines. Andrew *et al*. [1] proposes deep canonical correlation analysis (DCCA), where each input view is passed through a separate deep network (implementing a non-linear transformation), and the networks are jointly trained so that the their outputs are maximally correlated. In contrast to these works, our joint embedding is learned discriminatively using a maximum-margin cost. In addition, our embedding is loosely coupled, i.e., the image and pose embeddings do not explicitly share latent variables (layers). Rather the two embeddings are optimized through the dot-product similarity and supervised cost function, similar to learning a kernel embedding.

3. Maximum-margin structured learning

Our goal is to learn a score network that can assign maximum score to correct image-pose pairs and low scores to other pairs. The network structure is illustrated in Figure 1. Our network consists of two main components: an image feature extraction sub-network and an image-pose embedding sub-network. For the first sub-network, a CNN extracts high-level image features from the raw image. For the second sub-network, the image features and pose (3D joint coordinates) are separately fed through fully-connected layers, mapping them into two embedding spaces. The score function is then the dot-product between the two embeddings. Although the image/pose embeddings are calculated from separate sub-networks, training the full network will align the image/pose embeddings into a joint space, such that their dot-product is a suitable score function.

To train the network, we use a maximum-margin cost function that forces the score of the ground-truth image-pose pair to be larger than other image-pose pairs by at least a margin. We use a re-scaling margin, which is a function of the distance between the ground-truth pose and the other pose. In order to encourage image features that preserve pose information, we add an auxiliary task consisting of 3D body-joint prediction during training.

In the following, we use x to represent the image input, y as the ground-truth matching pose (3D joint coordinates), $\mathcal{Y}$ as the pose space, and θ as the network parameters.

3.1. Image feature extraction

The goal of the image extraction sub-network is to convert the raw input image to a more compact representation with pose information preserved. We use a deep CNN, consisting of 3 sets of convolution and max-pooling layers, to extract image features from the image. We use rectified linear units (ReLU) [20] as the activation function in the first 2 layers, and the linear activation function in the 3rd layer.

The outputs of the pooling layers is a set of feature maps, denoted as $\text{conv}^j(x)$, where j is the layer number. Each feature in the map has a receptive field in the input image, with higher layer features having larger receptive fields. Intuitively, the higher layer features will contain global information about the pose, which would be useful for dis-

tinguishing between grossly different poses. On the other hand, the lower layer features contain more detailed information about the pose, which will be helpful in distinguishing between similar poses.

3.2. Image-pose embedding

The image and pose inputs are in different spaces, and the goal of the image-pose embedding sub-network is to project the image features and the 3D pose into a joint embedding space where they can be compared effectively. The architecture of image and pose embedding network is shown in Figure 1. Inspired by [29, 19], we use features from both the middle- and top-convolutional layers. The middle- and top-layer features are each passed through separate fully connected layers, and then concatenated and passed through two more fully connected layers to form the image embedding $f_I(x)$. Specifically,

$$f_I(x) = h_4(h_3(\begin{bmatrix} h_1(\text{conv}^2(x)) \\ h_2(\text{conv}^3(x)) \end{bmatrix})), \tag{2}$$

where the activation function $h_i(x) = \text{ReLU}(W_i^T x + b_i)$ is a rectified linear unit with weight matrix W_i and bias b_i.

The input pose y is represented by the 3D coordinates of the body-joint locations, the dimensions of which are strongly correlated due the dependencies among joints. The pose is mapped into a non-linear embedding, so that it can be more easily combined with the image embedding. We use 2 fully connected layers for this transformation,

$$f_J(y) = h_6(h_5(y)). \tag{3}$$

3.3. Score prediction

We represent the score function between the image and pose inputs $f_S(x, y)$ as the inner-product between the image embedding $f_I(x)$ and pose embedding $f_J(y)$, i.e.,

$$f_S(x, y) = \langle f_I(x), f_J(y) \rangle. \tag{4}$$

One advantage of using inner-product is that the corresponding dimensions of the image/pose embedding vectors interact directly, which makes aligning the two embeddings easier. Another advantage is that it is very efficient to calculate. The calculation of the pose embedding does not depend on the image features, which means it can be calculated offline if the set of candidate poses is fixed.

Training the network will map the image and pose into similar embedding spaces, where their dot-product similarity serves as a suitable score function. This can be loosely interpreted as learning a multi-view "kernel" function, where the "high-dimensional" feature space is the learned joint embedding.

Our score function can also be interpreted as a SSVM, where the joint features are the element-wise product between the learned image and pose embeddings,

$$f'_S(x, y) = \langle w, f_I(x) \circ f_J(y) \rangle \tag{5}$$

where $\circ$ indicates element-wise multiplication, and w is the SSVM weight vector. The equivalence is seen by noting that during network training the weights w can be absorbed into the embedding functions $\{f_I, f_J\}$. In our framework, these embedding functions are discriminatively trained.

3.4. Maximum margin cost

Inspired by maximum-margin structured SVM [33], we use a maximum margin cost to learn the score function. The maximum margin cost ensures that the difference between the scores of two input pairs is at least a particular value (i.e., the margin). Different from the standard SVMs, with structured-SVM can have a margin that changes values based on dissimilarity between the two input pairs.

Similar to the structured-SVM, we use the margin rescaling surrogate loss,

$$\mathcal{L}_M(x, y, \hat{y}) = \max(0, f_S(x, \hat{y}) + \Delta(\hat{y}, y) - f_S(x, y)), \tag{6}$$

where (x, y) is a training image-pose pair, $\Delta(y, y')$ is a non-negative margin function between two poses, and $\hat{y}$ is the pose that most violates the margin constraint[1],

$$\hat{y} = \underset{y' \in \mathcal{Y}}{\text{argmax}} f_S(x, y') + \Delta(y, y') - f_S(x, y). \tag{7}$$

Intuitively, a pose with a high predicted score, but that is far from the ground-truth pose, is more likely to be the most violated pose. For the margin function, we use the mean per joint error (MPJPE), i.e.,

$$\Delta(y, y') = \frac{1}{J} \sum_{j=1}^{J} \|y_j - y'_j\|, \tag{8}$$

where y_j indicates the 3D coordinates of j-th joint in pose y, and J is the number of body-joints.

When the loss function in (6) is zero, then the score of the ground-truth image-pose pair (x, y) is at least larger than the margin for all other image-pose pairs (x, y'),

$$f_S(x, y) \geq f_S(x, y') + \Delta(y', y), \forall y' \in \mathcal{Y}. \tag{9}$$

On the other hand, if (6) is greater than 0, then there exists at least one pose y' whose score $f(x, y')$ violates the margin.

3.5. Multi-task global cost function

Following [18, 19], in order to encourage the image embedding to preserve more pose information, we include an auxiliary training task of predicting the 3D pose. Specifically, we add a 3D pose prediction layer after the penultimate layer of the image embedding network,

$$f_P(x) = g_7(h_3), \tag{10}$$

where h_3 is the output of the penultimate layer of the image embedding, and $g_i(x) = \tanh(W_i^T x + b_i)$ is the $\tanh$ activation function. The cost function for the pose prediction

[1] Note that $\hat{y}$ depends on the input (x, y) and network parameters θ. To reduce clutter, we write $\hat{y}$ instead of $\hat{y}(x, y, \theta)$ when no confusion arises.

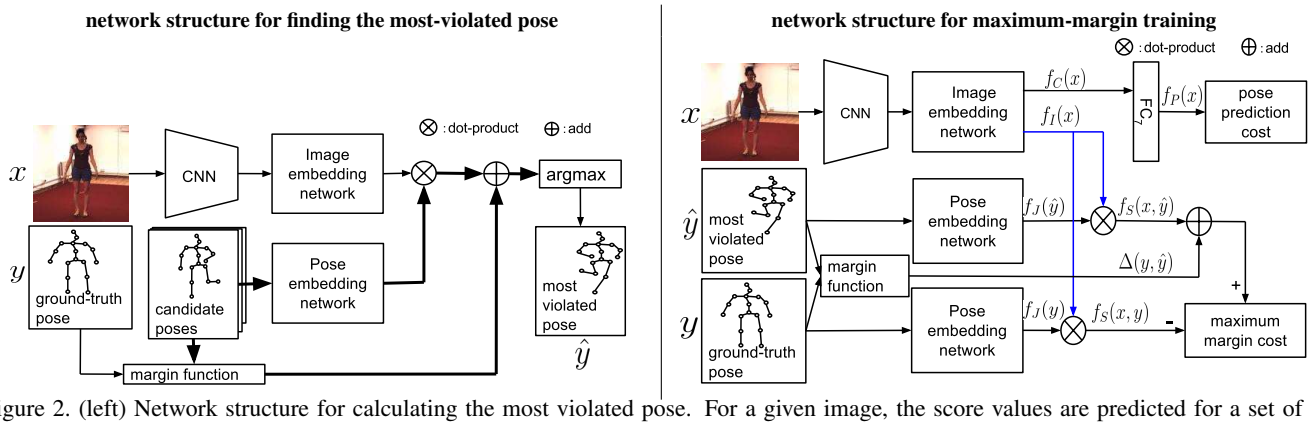

Figure 2. (left) Network structure for calculating the most violated pose. For a given image, the score values are predicted for a set of candidate poses. The re-scaling margin values are added, and the largest value is selected as the most-violated pose. Thick arrows represent an array of outputs, with each entry corresponding to one candidate pose. (right) Network structure for maximum-margin training. Given the most-violated pose, the margin cost and pose prediction cost are calculated, and the gradients are passed back through the network.

task is the square difference between the ground-truth pose and predicted pose,

$$\mathcal{L}_P(x, y) = \|f_P(x) - y\|^2. \tag{11}$$

Finally, given a training set of image-pose pairs $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$, our global cost function consists the structured maximum-margin cost, pose estimation cost, as well as a regularization term on the weight matrices,

$$\begin{aligned}\text{cost}(\theta) = &\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_M(x^{(i)}, y^{(i)}, \hat{y}^{(i)}) \\ &+ \frac{1}{N}\lambda\sum_{i=1}^{N}\mathcal{L}_P(x^{(i)}, y^{(i)}) + \alpha\sum_{j=1}^{7}\|W_j\|_F^2\end{aligned} \tag{12}$$

where i is the index for training samples, λ is the weighting for pose prediction error, α is the regularization parameter, and $\theta = \{(W_i, b_i)\}_{i=1}^7$ are the network parameters. Note that gradients from $\mathcal{L}_P$ only affect the CNN and high-level image features (FC_1-FC_3), and have no direct effect on the pose embedding network or image embedding layer (FC_4). Therefore, we can view the pose prediction cost as a regularization term for the image features. Figure 2 shows the overall network structure for calculating the max-margin cost function, as well as finding the most violated pose.

4. Training Algorithm

We use back-propagation [25] with stochastic gradient descent (SGD) to train the network. Similar to SSVM [15], our training procedure iterates between finding the most-violated poses and updating the network parameters:

1. Find the most-violated pose $\hat{y}$ for each training pair (x, y) using the pose selection network with current network parameters (Fig. 2 left);
2. Input $(x, y, \hat{y})$ into the max-margin training network (Fig. 2 right) and run back-prop to update parameters.

We call the tuple $(x, y, \hat{y})$ the extended training data. The training data is processed in mini-batches. We found that using momentum between mini-batches, which updates the parameters using the weighted average of the current gradient and previous update, always hinders convergence. This is because the maximum-margin cost selects different most-violated poses in each batch, which makes the gradient direction change rapidly between batches. To speed up the convergence of SGD, we use a line-search to find the best step-size for each mini-batch update. This was necessary because the the back-propagated gradients have high dynamic range, which stems from the cost function consisting of the difference between network outputs.

Although our score calculation is efficient, it is still computationally expensive to search the whole pose space to find the most-violated pose. Instead, we form a candidate set $\mathcal{Y}_\mathcal{B}$ for each mini-batch, and find the most-violated poses within the candidate set. The candidate set consists of C poses sampled from the pose space $\mathcal{Y}$. In addition, we observed that some poses are selected as the most-violated poses multiple times during training. Therefore, we also maintain a working set of most-violated poses, and include the top K most-frequent violated poses in the candidate set.

Our training procedure is summarized in Algorithm 1. Note that the selection of the most-violated pose from a candidate set, along with the back-propagation of the gradient for that pose, can be interpreted as a max-pooling operation over the candidate set.

5. Experiments

In this section, we evaluate our maximum margin structured learning network on human pose estimation dataset.

5.1. Dataset

We evaluate on the Human3.6M dataset [12], which contains around 3.6 million frames of video. The videos are recorded with four RGB camera, along with a MoCap sys-

Algorithm 1 Max-margin structured-network training

input: training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, pose space $\mathcal{Y}$, number of iterations M, number of mini-batches B, number of candidate poses C, number of most frequent violated poses K.
output: network parameters θ.
$\mathcal{V} = \emptyset$ {working set of most-violated poses}
for $t = 1$ **to** M **do** {loop over the whole training set}
  for $b = 1$ **to** B **do** {loop over mini-batches}
    $\mathcal{B}$ = ReadBatch()
    {get the current set of candidate poses $\mathcal{Y}_{\mathcal{B}}$}
    $\mathcal{Y}_{\mathcal{B}} = \text{UniformSample}(\mathcal{Y}, C)$ {get C poses}
    $\mathcal{Y}_{\mathcal{B}} = \mathcal{Y}_{\mathcal{B}} \cup \text{KMostFrequent}(\mathcal{V}, K)$
    {build the extended training data $\mathcal{D}$}
    $\mathcal{D} = \emptyset$
    for all $(x, y) \in \mathcal{B}$ **do**
      {calculate the most violated pose for (x, y)}
      $\hat{y} = \underset{y' \in \mathcal{Y}_{\mathcal{B}}}{\text{argmax}} \langle f_I(x), f_J(y') \rangle + \Delta(y, y')$
      $\mathcal{D} = \mathcal{D} \cup (x, y, \hat{y})$ {add to extended data}
      $\mathcal{V} = \mathcal{V} \cup \hat{y}$ {add to working set of violated poses}
    end for
    {update network parameters}
    StepSize = $\text{LineSearch}(\text{cost}, \mathcal{D}, \theta)$
    $\theta = \text{SGD}(\text{cost}, \mathcal{D}, \theta, \text{StepSize})$
  end for
end for

tem for measuring the joint positions. We treat the four RGB images separately, and project the MoCap coordinates to each camera coordinate system as the ground-truth pose.

As in [12, 18], the image input is a cropped image around the human. The training images are obtained by extracting a square image according to the bounding box provided in Human3.6M dataset [12], and resizing it to 128×128. As in [17], we augment the image training set by local translations and by adding random pixel noise during training. For local translations, a 112×112 sub-image is randomly selected from the training image. For pixel noise, random noise is added to all pixels according to the RGB covariance matrix calculated over the whole training set. The 3D pose input is a vector of the 3D coordinates of 17 body-joints.

5.2. Experiment setup

We follow the same protocol as in [18] for the training and test set – we use 5 subjects (S1, S5, S6, S7, S8) for training and validation, and 2 subjects (S9, S11) for testing.

Our structured-output network (denoted as StructNet) is trained using the algorithm from Section 4. Given a test image, ideally, the predicted pose should be found by searching the entire pose space $\mathcal{Y}$ for the pose with maximum score, as in (1). However, the pose space is continuous and exhaustive search is computationally intractable. Instead, we consider several approaches to approximate the search:

- StructNet-Max – the predicted pose is the pose in the training set with maximum score.
- StructNet-Avg(A) – since the training and test sets contain different subjects, the poses in the training set will not perfectly match the subjects in the test set. To allow for more pose variation, the predicted pose is the average of the A training poses with highest scores.
- StructNet-Avg(A)-APF – the problem with using StructNet-Avg is that the average pose is not guaranteed to be a valid pose. We use the annealing particle filtering (APF) [6] to generate a valid pose that best matches the pose estimated with StructNet-Avg(A). Specifically, APF adjusts the joint angles of a template pose to minimize the MPJPE with the StructNet-Avg pose. The template pose, which is a neutral "T" pose from the test subject, is initialized with the joint-angles from one of the top A poses. After APF converges, the joint-angles are converted into 3D joint coordinates.

The pose estimates on the test set are evaluated using MPJPE [12]. We also compare against multi-task deep networks (DconvMP-HML) [18], which trains a CNN using the pose prediction cost (Eq. 11), and LinKDE, the best performing method in [12].

5.3. Implementation details

The sizes of the network layers are shown in Figure 1. We follow the multi-task framework in [18] to initialize the weights for the convolutional layers. All the weight matrices for other layers are randomly initialized. When training the maximum-margin network, we fix the weights in the convolutional layers while still doing the data augmentation of the input image. The line-search was performed over the range $[10^{-7}, 10^{2}]$. We approximate the pose space $\mathcal{Y}$ with all the poses in the training set. The batch size is 128, and the size of the sampled candidate set is $C = 2000$. The number of most-frequent violated poses is $K = 10$. The weight for the auxiliary prediction task is $\lambda = 1$, and the regularization parameter is $\alpha = 0.0001$. We use dropout in the fully-connected layers $\{h_1, h_2\}$. The dropout rate is 75%. Our network is implemented in Theano [2].

5.4. Experiment results

Table 1 presents the MPJPE results on the test set for each action, as well as the overall average. We first compare the different methods for estimating the pose from StructNet. On all actions, StructNet-Avg (the average of the top scoring poses) yields better results than StructNet-Max (the maximum scoring pose), with overall reduction in error of about 10% when $A = 500$. Figure 3 plots the error versus different values of A. The error stabilizes between $A = 500$ and $A = 1000$, which represents ∼0.5% of the poses in the training set. Furthermore, applying APF to the average

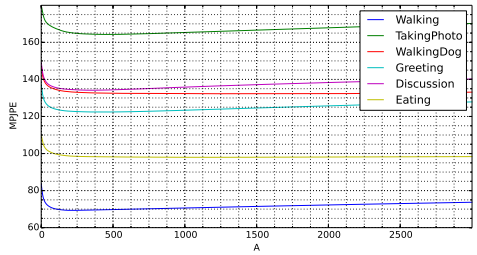

Figure 3. Pose error when averaging the top-A highest scoring training poses (StructNet-Avg(A)). $A = 500$ represents $\sim$0.5% of the poses in the training set.

pose from StructNet-Avg yields a valid pose with roughly the same MPJPE as StructNet-Avg.

Comparing to previous works, the error for StructNet-Avg is less than DconvMP-HML [18] and LinKDE [12] on all actions. The overall error is reduced 9.2% (from 133.54 for DconvMP-HML to 121.31 for StructNet-Avg(500)-APF). Also note that our method generates valid poses, whereas DconvMP-HML and LinKDE do not.

Next, we consider the role of the auxiliary pose prediction task in our network. We evaluate the performance of the auxiliary pose prediction on the test set (denoted as StructNet-Pred in Table 1). Overall, the performance of the auxiliary pose prediction task is similar to that of [18], which also uses a CNN for 3D pose estimation, but inferior to the poses obtained using the score function. We also test the effect of the auxiliary task on training the network. When removing the auxiliary task, i.e., $\lambda = 0$, the pose error increases (denoted as StructNet*-Max in Table 1). This demonstrates that the auxiliary task helps the network to converge to a good local optimum.

To justify the design choice of our pose embedding sub-network, we trained the whole network with different forms of pose embeddings: raw 3D joint coordinates, 1-layer network with fixed random weights, 1-layer network, and 2-layer network. The results are presented in Table 3. The network using no embedding (raw joint coordinates) has the highest error, while the 2-layer pose embedding has the lowest error, which suggests that embedding the pose in a suitable high-dimensional space is necessary.

Finally, to demonstrate robustness of our framework, we trained a network for each action category in Human3.6m (using the same network parameters), and evaluated on the online hidden test set[2]. The results are presented in Table 2. On average, the proposed framework achieves 8.8% lower error than LinKDE [12].

6. Visualization of image-pose embedding

In this section we visualize the latent features learned in the image-pose embedding. We first look at the 2 feature dimensions of the image embedding with the highest variance over all the training images. Figure 4a plots the values of these 2 features for each of the training images. To visualize the meaning of the features, in each local region, we show the average of the input images[3] corresponding to the feature points in that region. Figure 4b shows a similar plot for the same 2 feature dimensions in the pose embedding, with average poses over local regions of the space. The top-2 features in the embedding correspond to the orientation of the person. For example, in Figure 4a, the average image in the upper-part of the plot is a frontal view of the person, while the average image in the lower-part is the back view (similarly for the average poses in Figure 4b).

Embedding (dim.)	All
raw pose (51)	166.86 (92.63)
1-layer, random weights (1024)	145.95 (91.08)
1-layer (1024)	142.04 (87.98)
2-layer (1024)	135.63 (86.60)

Table 3. Comparison of different methods for pose embeddings.

Next, we look how the linear combination of embedding features encodes the abstract attributes of the person. We apply PCA on the image embedding vectors of all images in the training set, and then project the image embeddings onto two principal components. Figure 4c plots the two PCA coefficients using the same local region visualization as Figure 4a. Figure 4d shows the corresponding plot for the pose embedding. The first PCA component (x-axis in Figs. 4c and 4d) encodes the orientation (viewpoint) of the person, while the second PCA component (y-axis) encodes the attributes of the legs. For example, when the y-value is large the left leg is closer to the camera, while when the y-value is small, the right leg is closer to the camera.

Finally, these visualizations along with the supplemental video show that the learned embedding is smooth, even though the temporal order of frames are not used. We believe this is because the score function is learned using a max-margin constraint, which induces a topology of the embedding. Specifically, since the margin is based on the MPJPE between two poses, then the embedding vectors of any two poses should be at least as far apart (according to inner-product) as their MPJPE. In addition, the image and pose embeddings are properly aligned; 97% of the max-score poses for the training images are within 30 MPJPE of the ground-truth pose.

7. Conclusion

In this paper, we propose a maximum-margin structured learning framework with deep neural network for human pose estimation. Our framework takes image and pose as inputs and outputs a score value that represents a multi-view similarity between the two inputs (whether they depict the same pose). The network consists of a CNN for image feature extraction, and two separate sub-networks

[2]The action "Direction" is not included due to video corruption.

[3]For better visualization, we only use the images from a single subject.

Action	Walking	Discussion	Eating	Taking Photo	Walking Dog	Greeting	All
LinKDE(BS) [12]	97.07 (37.14)	183.09 (116.74)	132.50 (72.53)	206.45 (112.61)	177.84 (122.65)	162.27 (88.43)	162.25 (104.43)
DconvMP-HML [18]	77.60 (23.54)	148.79 (100.49)	104.01 (39.20)	189.08 (93.99)	146.59 (75.38)	127.17 (51.10)	133.54 (81.31)
StructNet-Max	83.64 (27.44)	149.09 (108.93)	109.93 (51.28)	179.92 (93.50)	147.24 (85.62)	136.90 (64.71)	135.63 (86.60)
StructNet-Avg(20)	75.01 (25.60)	140.90 (110.07)	104.10 (51.39)	173.26 (93.71)	139.47 (86.67)	129.08 (65.11)	128.11 (87.18)
StructNet-Avg(500)	69.75 (21.42)	134.37 (110.04)	98.19 (49.49)	164.28 (90.60)	132.53 (85.91)	122.44 (61.83)	121.46 (85.65)
StructNet-Avg(500)-APF	68.51 (22.21)	134.13 (112.87)	97.37 (51.12)	166.15 (92.95)	132.51 (87.37)	122.33 (64.56)	121.31 (87.95)
StructNet-Avg(1500)	71.46 (19.75)	137.18 (110.91)	98.01 (47.20)	166.62 (88.89)	132.26 (83.34)	124.58 (60.64)	123.04 (85.17)
StructNet-Avg(1500)-APF	69.97 (20.66)	136.88 (113.93)	96.94 (49.03)	168.68 (91.55)	132.17 (85.37)	124.74 (63.92)	122.85 (87.77)
StructNet-Pred	84.85 (24.17)	148.82 (102.63)	121.57 (50.47)	179.39 (83.72)	151.92 (76.26)	133.79 (56.16)	133.79 (56.16)
StructNet*-Max	87.15 (32.01)	161.62 (121.27)	119.50 (73.04)	196.24 (106.39)	154.91 (99.30)	145.30 (76.80)	145.74 (99.69)

Table 1. Results on Human3.6m: the MPJPE on the test set is calculated in millimeters (mm), with standard deviation parentheses.

Action	Discussion	Eating	Greeting	Phoning	Posing	Purchase	Sitting	SittingDown	Smoking	TakingPhoto	Waiting	Walking	WalkingDog	WalkingTogether	Avg
LinKDE(BS) [12]	108	91	129	104	130	134	135	200	117	195	132	115	162	156	133.81
DconvMP-HML [18]	103.11	91.68	108.38	109.49	116.45	145.24	145.14	329.96	110.35	174.97	112.43	99.16	153.29	116.44	136.47
StructNet-Avg(500)	92.97	76.70	98.16	92.70	106.86	140.94	135.46	260.75	98.03	170.83	105.11	99.40	138.53	109.30	122.03
StructNet-Avg(500)-APF	92.74	76.38	98.45	92.73	107.22	141.21	136.32	265.39	97.95	171.71	105.16	99.44	139.21	110.28	122.62

Table 2. Experimental results on the online (hidden) test set of Human3.6m.

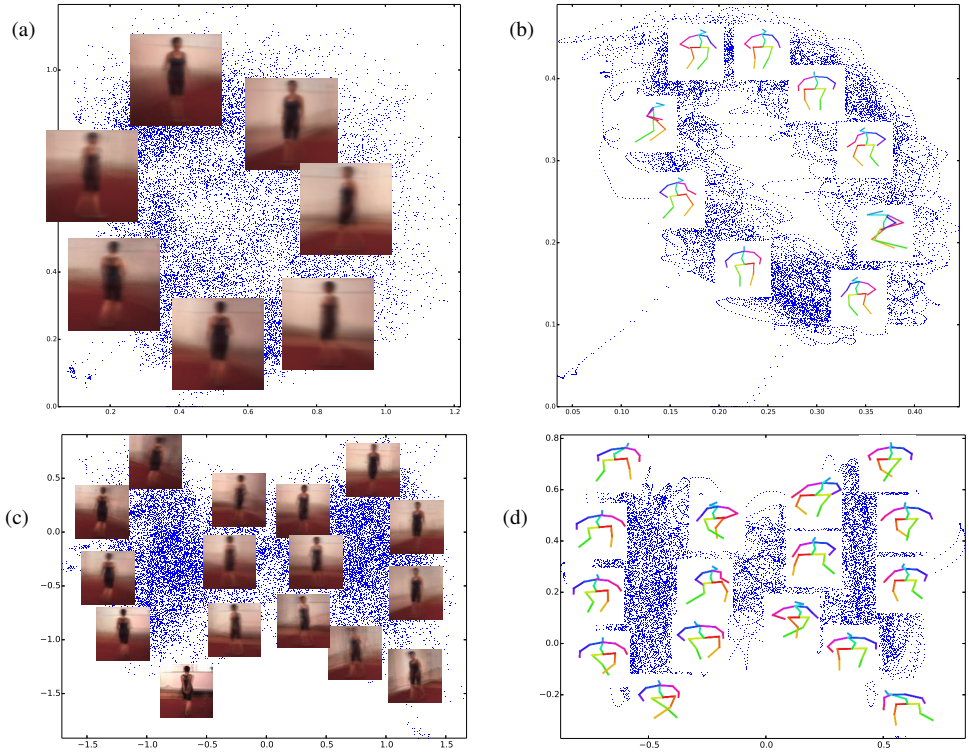

Figure 4. Visualizations of the learned image-pose embedding: (a) visualization of the two highest-variance features in the learned image embedding, and (b) the corresponding features in the pose embedding. (c) visualization of the two PCA coefficients of the learned image embedding and (d) pose embedding. In the pose plots, red/orange correspond to the right arm/leg, purple/green to the left arm/leg, and the cyan "nose" points in the forward direction of the person.

for non-linear transformation of the image and pose into a joint embedding, where the dot-product between the embeddings serves as the score function. We train the network using a maximum-margin cost function, which enforces a re-scaling margin between the score values of the ground-truth image-pose pair and other image-pose pairs. This specific form of embedding and score function makes inference computationally efficient, by allowing the pose embedding for a candidate set of poses to be calculated off-line. We evaluate our proposed framework on Human3.6M dataset and achieve significant improvement over the state-of-art. Finally, we show that the learned image-pose embedding encodes semantic attributes of the pose, such as the orientation of the person and the position of the legs. Our proposed framework is general, and future work will consider applying it to other structured-output tasks.

Acknowledgement. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 123212), and by a Strategic Research Grant from City University of Hong Kong (Project No. 7004417). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

[1] G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. In *ICML*, volume 28, pages 1247–1255, May 2013. 3

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. In *NIPS:Deep Learning and Unsupervised Feature Learning Workshop*, 2012. 6

[3] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *ICML*, pages 552–560, 2013. 2

[4] M. Burenius, J. Sullivan, and S. Carlsson. 3d pictorial structures for multiple view articulated pose estimation. In *CVPR*, pages 3618–3625, 2013. 2

[5] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*, 2014. 2

[6] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61(2):185–205, 2005. 6

[7] N. Dhungel, G. Carneiro, and A. P. Bradley. Deep structured learning for mass segmentation from mammograms. *CoRR*, abs/1410.7454, 2014. 2

[8] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *BMVC*, pages 1–11, 2009. 1, 2

[9] Felzenszwalb, P.F., Huttenlocher, and D.P. Pictorial structures for object recognition. *IJCV*, pages 55–79, 2005. 1

[10] C. Ionescu, L. Bo, and C. Sminchisescu. Structural SVM for visual localization and continuous state estimation. In *ICCV*, pages 1157–1164, 2009. 1

[11] C. Ionescu, F. Li, and C. Sminchisescu. Latent structured models for human pose estimation. In *ICCV*, pages 2220–2227, 2011. 1

[12] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE TPAMI*, 2014. 1, 5, 6, 7, 8

[13] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. *ICLR*, 2015. 2, 3

[14] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. In *ICLR*, April 2014. 1, 2

[15] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, Oct. 2009. 5

[16] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 1

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 6

[18] S. Li and A. B.Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, 2014. 1, 2, 4, 6, 7, 8

[19] S. Li, Z.-Q. Liu, and A. B. Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. *IJCV*, pages 1–18, 2014. 1, 2, 4

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[21] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, pages 689–696, 2011. 3

[22] M. Osadchy, Y. L. Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *J. Mach. Learn. Res.*, 8:1197–1215, May 2007. 2

[23] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR*, pages 512–519, 2014. 2

[24] J. A. Rodríguez and F. Perronnin. Label embedding for text recognition. In *BMVC*, 2013. 1, 2

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. In *Neurocomputing: Foundations of Research*, chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, 1988. 5

[26] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *In Proc. CVPR*, 2013. 1

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 2

[28] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2222–2230. Curran Associates, Inc., 2012. 3

[29] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *CVPR*. IEEE Computer Society, 2014. 4

[30] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 1, 2

[31] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1, 2

[32] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 1

[33] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, Dec. 2005. 4

[34] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 1