

# Chained Predictions Using Convolutional Neural Networks

Georgia Gkioxari<sup>1</sup>, Alexander Toshev<sup>2</sup>, and Navdeep Jaitly<sup>2</sup>

<sup>1</sup> University of California, Berkeley, USA  
gkioxari@eecs.berkeley.edu

<sup>2</sup> Google Inc., Mountain View, USA  
toshev@google.com, ndjaitly@google.com

**Abstract.** In this work, we present an adaptation of the sequence-to-sequence model for structured vision tasks. In this model, the output variables for a given input are predicted sequentially using neural networks. The prediction for each output variable depends not only on the input but also on the previously predicted output variables. The model is applied to spatial localization tasks and uses convolutional neural networks (CNNs) for processing input images and a multi-scale deconvolutional architecture for making spatial predictions at each step. We explore the impact of weight sharing with a recurrent connection matrix between consecutive predictions, and compare it to a formulation where these weights are not tied. Untied weights are particularly suited for problems with a fixed sized structure, where different classes of output are predicted at different steps. We show that chain models achieve top performing results on human pose estimation from images and videos.

**Keywords:** Structured tasks · Chain model · Human pose estimation

## 1 Introduction

Structured prediction methods have long been used for various vision tasks, such as segmentation, object detection and human pose estimation, to deal with complicated constraints and relationships between the different output variables predicted from an input image. For example, in human pose estimation the location of one body part is constrained by the locations of most of the other body parts. Conditional Random Fields, Latent Structural Support Vector Machines and related methods are popular examples of structured output prediction models that model dependencies among output variables.

A major drawback of such models is the need to hand-design the structure of the model in order to capture important problem-specific dependencies amongst the different output variables and at the same time allow for tractable inference. For the sake of efficiency, a specific form of conditional independence amongst output variables is often assumed. For example, in human pose estimation, a predefined kinematic body model is often used to assume that each body part is independent of all the others except for the ones it is attached to.



**Fig. 1.** A description of our model for the task of body pose estimation compared to pure feed forward nets. **Left:** Feed forward networks make independent predictions for all body parts simultaneously and fail to capture contextual cues for accurate predictions. **Right:** Body parts are predicted sequentially, given an image and all previously predicted parts. Here, we show the chain model for the prediction of *Right Wrist*, where predictions of all other joints in the sequence are used along with the image.

To alleviate some of the above modeling simplifications, structured prediction problems have been solved with sequential decision making, where all earlier predictions influence later predictions. The SEARN algorithm [1] introduced a very general formulation for this approach, and demonstrated its application to various natural language processing tasks using losses from binary classifiers. A related model recently introduced, the sequence-to-sequence model, has been applied to various sequence mapping tasks, such as machine translation, speech recognition and image caption generation [2-4]. In all these models the output is a sentence - where the words of the sentence are predicted in a first to last order. This model maximizes the log probability for output sequence conditioned on the input, by decomposing the probability of an output sequence with the multiplicative chain rule of probability; at each index of the output, the next prediction is made conditioned on all previous outputs and the input. A recurrent neural network is used at every step of the output and this allows parameter sharing across all the output steps.

In this paper we borrow ideas from the above sequence-to-sequence model and propose to extend it to more general structured outputs encountered in computer vision – human pose estimation from a single image and video. The contributions of this work are as follows:

- A *chain model for structured outputs*, such as human pose estimation. The body part locations are predicted sequentially, where the prediction of each body part is dependent on *all previously predicted* body parts (See Fig. 1). The model is formulated using a neural network in which the feature extraction and prediction models are learned end-to-end. Since we apply the model to spatial labelling tasks we use convolutional neural networks in both the inputs and outputs. The output convolutional neural networks is a multi-scale

deconvolution that we call *deception* because of its relationship to deconvolution [5,6] and inception models [7].

- We demonstrate *two formulations of the chain model* - one without weight sharing between different predictors (poses in images) to allow semantic-specific flow of information and the other with weight sharing to enforce recurrence in time (poses in videos). The latter model is a RNN similar to the sequence-to-sequence model.

The above model achieves top performing results on the MPII human pose dataset – 86.1% PCKh. We achieve state-of-the art performance for pose estimation on the PennAction video dataset – 91.8% PCK.

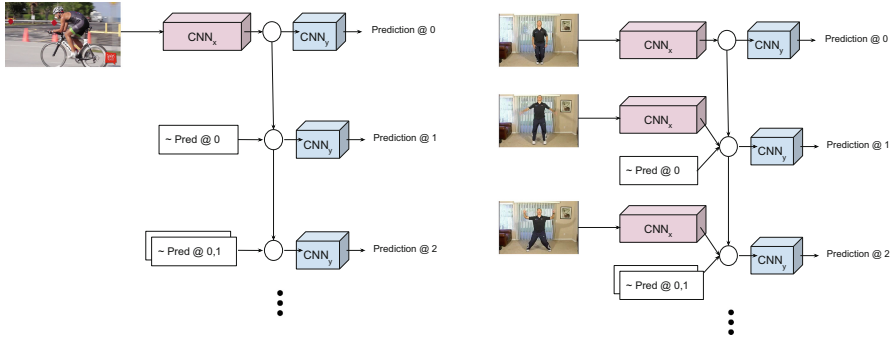
## 2 Related Work

*Structured output prediction as sequence prediction.* The use of sequential models for structured predictions is not new. The SEARN algorithm [1] laid down a broad framework for such models in which a sequence of actions is generated by conditioning the next action on previous actions and the data. The optimization method proposed in SEARN is based on iterative improvement over policies using reinforcement learning.

A similar class of models are the more recent sequence-to-sequence models [2,8] that map an input sequence to an output sequence of fixed vocabulary. The models produce output variables, one at a time, conditioned on inputs and previous output variables. A next-step loss function is computed at each step, using a recurrent neural network. Sequence-to-sequence models have been shown to be very effective at a variety of language tasks including machine translation [2], speech recognition [3], image captioning [4] and parsing [9]. In this paper we use the same idea of chaining predictions for structured prediction on two vision problems - human pose estimation in individual frames and in video sequences. However, as exemplified in the pose estimation case, since we have a fixed output structure we are not limited to using recurrent models.

In the pose prediction problem, we used a fixed ordering of joints, that is motivated by the kinematics of the human body. Prior work in sequential modelling has explored the idea of choosing the best ordering for a task [10–12]. For example, Vinyals et al. [10] explored this question and found that for some problems, such as geometric problems, choosing an intuitive ordering of the outputs results in slightly better performance. However for simpler problems most orderings were able to perform equally well. For our problem, the number of joints being predicted is small, and tree based ordering of joints from head to torso to the extremities seems to be the intuitively correct ordering.

*Human pose estimation.* Human pose estimation has been one of the major playgrounds for structured prediction models in computer vision. Historically, most of the research has focused on graphical models, starting with tree-based decompositions [13–16] motivated by kinematic models of the human body.



**Fig. 2.** A visualization of our chain model. **Left:** single image case. **Right:** video case. In both cases, an image is encoded with a CNN ( $CNN_x$ ). At each step, the previous output variables are combined with the hidden state, through the sequential modules. A CNN decoder ( $CNN_y$ ) makes predictions each step,  $t$ . There are two differences between the two cases: (i) for video  $CNN_x$  receives at each step a frame as an input, while for single image there is no such input; (ii) for video  $CNN_y$  share parameters across steps, while for single image the parameters are untied.

Many of these models assume conditional independence of a body part from all other parts except the parent part as defined by the kinematic body model (see pictorial structure model [13]). This simplification comes at a performance cost and has been addressed in various ways: mixture model of parts [17]; mixtures of full body models [18]; higher-order spatial relationships [19]; image dependent pictorial structures [20–23]. Like these above approaches, we assume an order among the body parts. However, this ordering is used only to decompose the joint probability of the output joints into a particular ordering of variables in the chain rule of probability, and not to make assumptions about the structure of the probability distribution. Because no simplifying assumptions are made about the joint distribution of the output variables it leads to a more expressive model, as exemplified in the experimental section. The model is only constrained by the ability of neural networks to model the conditional probability distributions that arise from the particular ordering of the variables chosen. In addition, the correlations among parts are learned through a set of non-linear operations instead of imposing binary term constraints on hand-designed image features (e.g. RGB values, location) as done in CRFs.

It is worth noting that there have been models for pose estimation where parts are sequentially refined [24–27]. In these models an initial prediction is made of all the parts; in subsequent steps, all part predictions are refined based on the image and earlier part predictions. However, note that the predictions are initially independent of each other.

### 3 Chain Models for Structured Tasks

Chain models exploit the structure of the tasks they are designed to tackle by sequentially predicting their outputs. To capture this structure each output prediction is conditioned on all outputs predicted already. This philosophy has been exploited in language processing where sentences, expressed as word sequences, need to be predicted [2, 8] from inputs. In recent automatic image captioning work [4, 28], for example, a sentence  $Y$  is generated from an image  $X$  by maximizing the likelihood  $P(Y | X)$ . The chain rule is applied, consecutively to model each output  $Y_t$  (here a word) given the image  $X$  and all the previous outputs  $Y_{<t}$  in the output sequence.

In computer vision, recognition problems, such as segmentation, detection and pose estimation, demonstrate rich structure with complex dependencies. In this work, we model this structure with a simple and efficient recognition machine that makes little to no assumptions about the structure, other than the ability of a neural network to model complex, incremental conditional distributions.

Mathematically, let  $Y = \{Y_t\}_{t=0}^{T-1}$  be the  $T$  objects to be detected. For example, for the pose prediction problem,  $Y_t$  is the location of the  $t$ -th body part. In video prediction problems,  $Y_t$  is the location of an object in the  $t$ -th frame of a video. Using the chain rule we decompose  $P(Y = y | X)$  as follows:

$$P(Y = y | X) = P(Y_0 = y_0 | X) \prod_{t=1}^{T-1} P(Y_t = y_t | X, y_0, \dots, y_{t-1}) \quad (1)$$

From the above equation, we see that the likelihood of assigning value  $y_t$  to the  $t$ -th variable is given by  $P(Y_t = y_t | X, y_0, \dots, y_{t-1})$ , and depends on both the input  $X$  as well as the assignment of previous variables. In this work, we model the likelihood  $P(Y_t = y_t | X, y_0, \dots, y_{t-1})$  with a convolutional neural network (CNN). The direct dependence of the current prediction on the ground truth values of previous variables allows for the model to capture all necessary relationships without making any assumption about the joint distributions of all the variables, other than assuming that each successive conditional distribution,  $P(Y_t = y_t | X, y_0, \dots, y_{t-1})$ , can be computed with a neural network.

#### 3.1 Chain Models for Single Images

In the case of single images, the input  $X$  is the image while the  $t$ -th variable  $Y_t$  can be, for example, the location of the  $t$ -th object in image  $X$  (see Fig. 2).

The probability of each step in the decomposition of Eq. (1) is defined through a hidden state  $h_t$  at step  $t$ , which carries information about the input as well as states at previous steps. In addition it incorporates the values  $y_{<t}$  from previous steps. The final probability for variable  $Y_t$  is computed from the hidden state:

$$h_t = \sigma(w_t^h * h_{t-1} + \sum_{i=0}^{t-1} w_{i,t}^y * e(y_i)) \quad (2)$$

$$P(Y_t = y_t | X, y_0, \dots, y_{t-1}) = \text{Softmax}(m_t(h_t)) \quad (3)$$

In the above equation, the previous variables are first transformed through a full neural net  $e(\cdot)$ . Parameters  $w_t^h$  and  $w_{i,t}^y$  then linearly transform the previous hidden state and a function of previous output variables,  $e(\cdot)$ , and a non-linearity  $\sigma$  is then applied to each dimension of this output. The nonlinearity  $\sigma$  of choice is a Rectified Linear Unit. Finally,  $*$  denotes multiplication. In image applications, however, the hidden state  $h$  can be a feature map and the prediction  $y$  a location in the image. In such cases,  $*$  denotes convolution and  $e$  is a CNN. Note that, as long as we feed in just the last variable  $y_{t-1}$  in this equation, the recurrent equation insures that we condition on the entire history of joints. However feeding in more of the previous joints makes it easier for the model to learn the conditional distributions directly. In the computation of the conditional probability of  $y_t$  from  $h_t$  we use another neural net  $m_t$ , which produces scores for potential object location. By applying a softmax function over these scores we convert them to a probability distribution over locations.

The initial state  $h_0$  is computed based solely on the input  $X$ :  $h_0 = \text{CNN}(X)$ .

This formulation is reminiscent of recurrent networks (RNNs), the equations define how to transform a state from one step to the next. We differ, however, from RNNs in one important aspect, the parameters in Eqs. (2-3) are not necessarily tied. Indeed, parameters  $w_t^h$  and  $w_{i,t}^y$  are indexed by the step. This design choice is appropriate for tasks such as human pose estimation where the number of outputs  $T$  is fixed and where each step is different from the rest. In other applications, e.g. video, we tie these parameters:  $w_t^h = w_0^h$  and  $w_{i,t}^y = w_{i,0}^y, \forall i, t$ .

### 3.2 Chain Models for Videos

For videos, the input is a sequence of images  $X = \{X_t\}_{t=0}^{T-1}$  (Fig. 2). Predictions are made at each step, as the images are fed in. At each step  $t$ , we make predictions for the image  $X_t$  at that step, using the past images, and the past output variables. Thus, we modify the equation for the hidden state as follows:

$$h_t = \sigma(w_t^h * h_{t-1} + \text{CNN}(X_t) + \sum_{i=t-T_H}^{t-1} w_{t-i,t}^y * e(y_i)) \tag{4}$$

where we add features extracted from image  $X_t$  using a CNN. The final probability is computed as in Eq. (3).

In videos we often need to predict the same type of information at each step, e.g. location of all body joints of the person in the current frame. As such, the predictors can have the same weights. Thus, we tie the parameters  $w_t^h$ ,  $w_{i,t}^y$ , and  $m_t$  together, which results in a convolutional RNN.

As before, the connections from hidden state at the previous step guarantees that the prediction at each time step uses output variables from all previous steps, as long as the previous output variable  $Y_{t-1}$  is fed in at time  $t$ . However, feeding in a larger time horizon  $T_H$  leads to an easier learning problem.

### 3.3 Improved Learning with Scheduled Sampling

So far, we have described the method as using the input and only ground truth values of the previous output variables when making a prediction for the next output variable. However, it has previously been observed that for sequence-to-sequence models overfitting can be mitigated by probabilistically substituting ground truth values of previous output variables with samples from the probability distribution predicted by the model [29]. One challenge that arises in this is that, at the start of the training, the predicted probability distributions are wildly inaccurate and thus, feeding in samples from the distribution is counter-productive. The authors of [29] propose a method, called *scheduled sampling*, that uses an annealing schedule that feeds in only the ground truth outputs at the start of the training and increases the rate of sampling from the predictions of the model towards the end of the training. We use the idea of scheduled sampling in our paper and find that it leads to improved results.

## 4 Experimental Evaluation

To evaluate the proposed model, we apply it on human pose estimation, which is challenging and of great interest due to the complex relationship among body parts. In the single image case, we use the chain model to capture the structure of pose in space, i.e. how the location of a part influences others. For the videos, our model captures the constraints and dynamics of the body pose in time.

**Tasks and Datasets.** For our single image experiments we use the MPII Human Pose dataset [30], which consists of about 40 K instances of people performing various actions. All frames come with a maximum of 16 annotated joints (e.g. *Top Head*, *Right Ankle*, *Left Knee*, etc.). For the task of pose estimation in video we use the Penn Action dataset [31], which consists of 2326 video sequences of people performing various sports. All frames come with a maximum of 13 annotated joints. During evaluation, if a joint prediction lies within a predefined distance, proportional to the size of the person, from the ground truth location it is counted as a correct detection. This metric is called PCK [30, 32].

Our model is illustrated in Fig. 2. We experiment with two choices for  $\text{CNN}_x$ , the network which encodes the input image. First, a shallow CNN which consists of six layers each followed by a rectified linear unit [33] and Batch Normalization [34]. The first four layers include max pooling with stride 2, leading to an effective stride of 16. This network is described in Fig. 3. Second, we experiment with a deeper network of identical architecture to inception-v3 [35]. We discard the last convolutional layer of inception-v3 and connect the output to  $\text{CNN}_y$ .

The  $\text{CNN}_y$  network decodes the hidden state to a heatmap over possible locations of a single body part. This heatmap is converted to a probability distribution over locations using a softmax. The network consists of two towers of deconvolutional layers each of which increases the width and height of the feature maps by a factor of 2. Note that the deconvolutional towers are

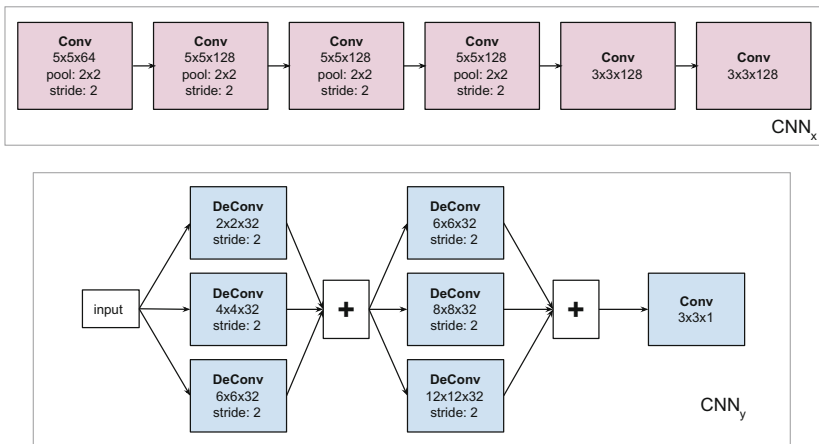
multi-scale - in one layer, different filter sizes are used and combined together. This is similar to the inception model [7], with the difference that here it is applied with the deconvolution operation, and hence we call it *deception*.

### 4.1 Pose Estimation from a Single Image

In this application case, we use the chain model to predict the joints sequentially. The sequence with which the joints are processed is fixed and is motivated by the marginal distributions of the joints. In particular, we sort the joints in descending order according to the detection rates of an unchained feed forward net. This allows for the easy cases to be processed first (e.g. *Torso, Head*) while the harder cases (e.g. *Wrist, Ankle*) are processed last, and as a result use the contextual information from the joints predicted before them.

**Inference.** At test time, we use beam search to infer the optimal location of the joints. Note that exact inference is infeasible, due to the size of the search space (a total of  $(HW)^T$  possible solutions, where  $H \times W$  is the size of the prediction heatmap and  $T$  are the number of joints). At each step  $t$ , the best  $B$  predictions are stored, where each prediction is the sequence of the first  $t$  joints. The quality of a full body pose prediction is measured by its log-probability, which is the sum of the log-probabilities corresponding to the individual joint predictions.

An exact implementation of chain rule conditions on predictions made at every step. Alternatively, one could skip the non-differentiable sampling operation and use the probability distributions directly. Even though this is not an



**Fig. 3.** Description of the components of our network,  $CNN_x$  and  $CNN_y$ . Each box represents a convolutional or deconvolutional layer, where  $w \times h \times f$  denotes the width  $w$ , the height  $h$  of the filters and  $f$  denotes the number of filters. In each layer the filter is applied with stride 1 if not noted otherwise. Finally, in each layer after the filtering operation a ReLU and batch normalization are applied.



exact application of the chain rule, it allows for the gradients to flow back to the output of each task. We found that this approximation led to very similar performance - it slowed down training time by a factor of 3 and sped up inference by a factor of  $B$ .

**Learning Details.** We use an SGD solver with momentum to learn the model parameters by optimizing the loss. The loss for one image  $X$  is defined as the sum of losses for individual joints. The loss for the  $k$ -th joint is the cross entropy between the predicted probability  $P_k$  over locations of the joint and the ground-truth probability  $P_k^{\text{gt}}$ . The former is defined based on the heatmap  $h_k$  output by  $\text{CNN}_y$  for the  $k$ -th joint:  $P_k(x, y) = \frac{e^{h_k(x, y)}}{\sum_{(x', y')} e^{h_k(x', y' )}}$ . The latter is defined based on a distance  $r$  - all locations within radius  $r$  of the ground-truth joint location are assigned same nonzero probability  $P_k^{\text{gt}}(x, y) = 1/N$ , all other locations are assigned probability 0.  $N$  is a normalizer guaranteeing  $P_k^{\text{gt}}$  is a probability.

**Table 1.** PCKh performance on the MPII validation set. Rows 1 and 2 show results for 9-layered CNN models, with multi-scale (deception) and single scale deconvolutions. Row 3 show results for a 24-layer model with deception, but without chained outputs. Row 4 shows results for our chain model with comparable depth and number of parameters as the 24-layer model, but with chained predictions. We observe clear improvement over the baselines. The performance is further improved using multiple crops of the input at test time, at row 5. Row 6 shows the performance of the oracle, where the correct values of previous output is fed into the network at each step. Row 7 and 8 show the performance for a base and chain model when inception-v3, pre trained on ImageNet, is used as the encoder network. Using a deeper architecture leads to substantially improved results across all joints.

| PCKh (%)                   | Torso | Head | Shldr | Elbow | Wrist | Hip  | Knee | Ankle | Mean |
|----------------------------|-------|------|-------|-------|-------|------|------|-------|------|
| Base Network               | 86.8  | 91.9 | 85.8  | 74.5  | 69.0  | 71.1 | 61.4 | 50.6  | 73.9 |
| Base Net. w/single deconv. | 86.0  | 91.7 | 85.1  | 72.9  | 68.0  | 69.4 | 59.7 | 48.5  | 72.6 |
| Very Deep Base Network     | 88.1  | 92.0 | 86.1  | 74.1  | 67.7  | 73.7 | 64.7 | 58.0  | 75.6 |
| Chain Model                | 86.8  | 93.2 | 88.3  | 79.4  | 74.6  | 77.8 | 71.4 | 65.2  | 79.6 |
| Chain Model w/multi-crop   | 88.7  | 94.4 | 90.0  | 82.6  | 78.6  | 80.2 | 74.8 | 68.4  | 82.2 |
| Oracle Chain Model         | 87.2  | 95.9 | 93.4  | 83.3  | 82.3  | 95.2 | 77.6 | 72.3  | 85.9 |
| Inception Base Network     | 91.1  | 95.0 | 90.2  | 81.0  | 77.4  | 77.2 | 73.7 | 64.6  | 81.3 |
| Inception Chain Model      | 91.7  | 95.7 | 92.2  | 85.3  | 82.2  | 82.9 | 80.0 | 72.4  | 85.3 |

The final loss for  $X$  reads as follows:

$$L(\{h_k\}_{k=0}^{T-1}) = \sum_{k=0}^{T-1} \sum_{(x,y)} P_k^{\text{gt}}(x, y) \log P_k(x, y) \quad (5)$$

We use batch size of 16; initial learning rate of 0.003 that was decayed every 100 K steps (50 K for the inception model); radius of  $r = 0.01 \times (W + H)/2$ . The model was trained for 120 K iterations (55 K for the inception model). Our images are rescaled to  $224 \times 224$  ( $299 \times 299$  for the inception model). The weights of the network are initialized by sampling from a normal distribution of zero mean and 0.01 standard deviation. For the inception model, we initialize the weights of  $\text{CNN}_x$  with weights from an ImageNet model.

**Results.** Table 1 shows the PCKh performance on the MPII validation set of our chain model and our baseline variants.

Rows 1, 2 & 3 show the performance of pure feed forward networks for the task in question. The 1st row shows the performance of a 9-layer network, shallow  $\text{CNN}_x + \text{CNN}_y$ , which we call base network. The 2nd row is a similar network, where each deconvolutional tower, which we call *deception*, in  $\text{CNN}_y$  is replaced by a single deconvolution. The difference in performance shows that multi-scale deconvolutions lead to a better and very competitive baseline. Finally, the 3rd row shows the performance of a very deep network consisting of 24 layers. This network has the same number of parameters and the same depth as our chain model and serves as the baseline which we improve upon using the chain model.

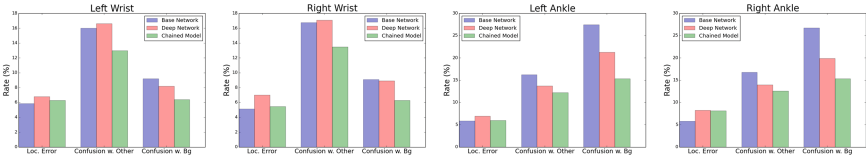
Row 4 shows the performance of our chain model. This model improves significantly over all the baselines. The biggest gains are observed for *Wrists* and *Ankles*, which is a clear indication that conditioning on the predictions of previous joints provides cues for better localization.

Row 5 shows the performance of the chain model with multi-crop evaluation, where at test time we average the predictions from flipping and jittering of the input image.

Row 6 shows the performance of an *oracle* chain model. For this model, at each step  $t$  we use the oracle (ground truth) locations of all previous joints. This model is an estimate of the upper bound performance of our chain model, as it predicts the location of a joint given perfect knowledge of the location of all other joints which precede it in the sequence.

Row 7 shows the performance of the inception base network,  $\text{CNN}_x + \text{CNN}_y$ , where  $\text{CNN}_x$  is the inception-v3 [35]. We observe significant gains when using the inception-v3 architecture compared to a shallower 6-layer network for the encoder network, at the expense of more computations.

Row 8 shows the performance of the inception chain model. For both the inception base and chain model we use multi-crop evaluation. In both cases, the inception-v3 parameters were initialized with weights from an ImageNet model. The inception chain model leads to significant gains compared to its base network (row 7). The improvements are more evident for the joints of *Wrist*, *Knee*, *Ankle*.



**Fig. 4.** Error analysis of the predictions made by the base network (blue), the very deep model (red) and our chain model (green), for *Wrist* and *Ankle*. Each figure shows the error rates, categorized in three classes, localization error, confusion with other joints and confusion with the background. (Color figure online)

**Error Analysis.** Digging deeper into the models, we perform an error analysis for the base network  $CNN_x + CNN_y$ , the very deep network and our chain model. For this analysis, the 6-layer encoder network  $CNN_x$  is used for all models. Similar to [36], we categorize the erroneous predictions into the three distinct classes: (a) localization error, i.e. the prediction is within  $[\alpha, \beta] \times HeadSize$  of the true location, (b) confusion with other joints, i.e. the prediction is within  $\alpha \times HeadSize$  of a different joint, and (c) confusion with the background, i.e. the prediction lies somewhere else in the image. According to PCKh, a prediction is correct if it falls within  $0.3 \times HeadSize$ . We set  $\beta = 0.5$ .

Figure 4 shows the error analysis for the hardest joints, namely *Wrist* and *Ankle*. Each plot consists of three sets of bars, the rates for error localization, confusion with other joints and confusion with background. According to the plots, the chain model reduces the misses due to confusion with other joints and the background. For *Wrists*, the confusion with other joints is the dominating error mode, and further analysis shows that the main source of confusion comes mainly from the opposite wrist and then the nearby joints. For *Ankles*, the biggest error mode comes from confusion with the background, which is not surprising since lower legs are usually heavily occluded and lack strong appearance cues.

Figure 5 shows some examples of our predictions on the MPII dataset.



**Fig. 5.** Examples of predictions by our chain model on the MPII dataset.

**Table 2.** Performance on the MPII test set. A comparison of our chain model, with a shallow 6 layer and an inception-v3 encoder, with leading approaches in the field.

| Method                 | Head | Shoulder | Elbow | Wrist | Hip  | Knee | Ankle | Total |
|------------------------|------|----------|-------|-------|------|------|-------|-------|
| Carreira et al. [26]   | 95.7 | 91.7     | 81.7  | 72.4  | 82.8 | 73.2 | 66.4  | 81.3  |
| Tompson et al. [38]    | 96.1 | 91.9     | 83.9  | 77.8  | 80.9 | 72.3 | 64.8  | 82.0  |
| Hu and Ramanan [39]    | 95.0 | 91.6     | 83.0  | 76.6  | 81.9 | 74.5 | 69.5  | 82.4  |
| Pishchulin et al. [40] | 94.1 | 90.2     | 83.4  | 77.3  | 82.6 | 75.7 | 68.6  | 82.4  |
| Lifshitz et al. [41]   | 97.8 | 93.3     | 85.7  | 80.4  | 85.3 | 76.6 | 70.2  | 85.0  |
| Wei et al. [27]        | 97.8 | 95.0     | 88.7  | 84.0  | 88.4 | 82.8 | 79.4  | 88.5  |
| Newell et al. [37]     | 97.6 | 95.4     | 90.0  | 85.2  | 88.7 | 85.0 | 80.6  | 89.4  |
| Chain model            | 93.8 | 91.8     | 84.2  | 79.4  | 84.4 | 77.9 | 70.7  | 84.1  |
| Inception Chain Model  | 97.9 | 93.2     | 86.7  | 82.1  | 85.2 | 81.5 | 74.0  | 86.1  |

**Comparison to Other Approaches.** We evaluate our approach on the MPII test set and compare to other methods on the task of pose estimation from a single image. Table 2 shows the results of our approach and other leading methods in the field. We show the performance of both versions of our chain model, using a shallow 6-layer encoder as well as the inception-v3 architecture. For the shallow chain model, we ensemble two chain models trained at different input scales. For the inception chain model, no ensembling was performed.

The leading approaches by Wei *et al.* [27] and Newell *et al.* [37] rely on iteratively refining predictions. In particular, predictions are made initially for all joints independently. These predictions, which are quite poor (see [27]), are fed subsequently into a network for further refinement. Our approach produces only one set of predictions via a single chain model and does not refine them further. One could combine the two ideas, the one of chained predictions and the one of iterative refinement, to achieve better results.

## 4.2 Pose Estimation from Videos

Our chain models in time are described in Eq. 4 and illustrated in Fig. 2. Here, the task is to localize body parts in time across video frames. The output variables from the joints of the previous frames are used as inputs to make a prediction for the joints in the current frame. We apply the chaining in two different ways - first, only in time, where each joint is predicted independently of the other joints (as in our baseline models), but chaining is done in time, and second, with chaining both in time and in joints.

**Pose Estimation in Time.** As shown in Fig. 2, the chain model sequentially processes the video frames. The predictions at the previous time steps are used through a recurrent module in order to make a prediction at the current time step. Again, we use a heatmap to encode the location of a part in the frame.

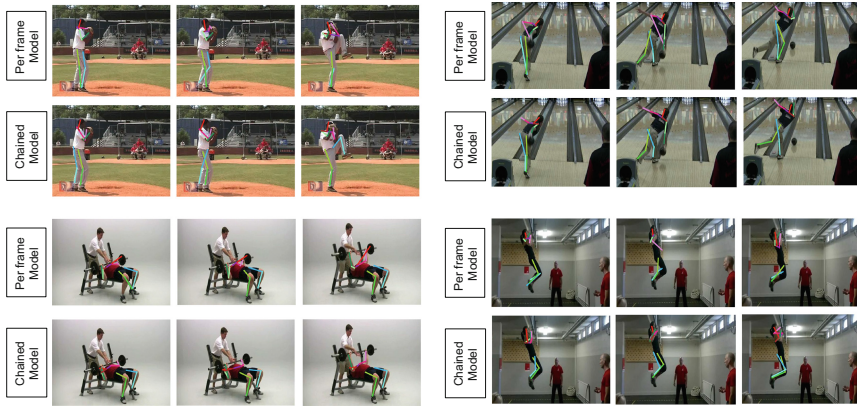
**Table 3.** PCK performance on the Penn Action test set. We show the performance of our chain model for two choices of the time horizon  $T_H$  and compare against the per-frame model, with and without temporal smoothing, and a baseline convolutional RNN model. The chain model with  $T_H = 3$  improves the localization accuracy across all joints. The method by Nie *et al.* [42] is shown for comparison.

| PCK (%)                                    | Head | Shldr | Elbow | Wrist | Hip  | Knee | Ankle | Mean |
|--|------|-------|-------|-------|------|------|-------|------|
| Nie <i>et al.</i> [42]                     | 64.2 | 55.4  | 33.8  | 24.4  | 56.4 | 54.1 | 48.0  | 48.0 |
| Base Network                               | 94.1 | 90.3  | 84.2  | 83.5  | 88.7 | 87.2 | 87.7  | 87.5 |
| Base Network<br>w/smoothing                | 93.1 | 91.8  | 85.7  | 78.8  | 90.2 | 91.9 | 91.1  | 88.6 |
| RNN  | 95.3 | 92.5  | 87.9  | 87.5  | 91.1 | 89.8 | 90.1  | 90.1 |
| Chain Model, $T_H = 1$                     | 95.8 | 93.2  | 88.9  | 89.6  | 91.3 | 89.8 | 91.2  | 91.0 |
| Chain Model, $T_H = 3$                     | 95.8 | 94.1  | 90.0  | 90.2  | 91.3 | 90.6 | 91.8  | 91.7 |
| Chain Model in time &<br>joints, $T_H = 3$ | 95.6 | 93.8  | 90.4  | 90.7  | 91.8 | 90.8 | 91.5  | 91.8 |

The details of our learning procedure are identical to the ones described for the single image case. The only difference is that each training example is now a sequence of images  $X = \{X_t\}_{t=0}^{T-1}$  each of which has a ground-truth pose. Thus, the loss for  $X$  is the sum over the losses for each frame. Each frame loss is defined as in the case of single image (see Eq. (5)).

We train our model for 120K iterations using SGD with momentum of 0.9, a batch size of 6 and a learning rate of 0.003 with step decay 100K. Images are rescaled to  $256 \times 256$ . A relative radius of  $r = 0.03$  is used for the loss. The weights are initialized randomly from a normal distribution with zero mean and standard deviation of 0.01.

Table 3 shows the performance on the Penn Action test set. For consistency with previous work on the dataset [42], a prediction is considered correct if it lies within  $0.2 \times \max(s_h, s_w)$ , where  $s_h, s_w$  is the height and width, respectively, of the instance in question. We refer to this metric as PCK. (Note that this is a weaker criterion than the one used on the MPII dataset). We show the per frame performance, as produced by a base network  $\text{CNN}_x + \text{CNN}_y$  trained to predict the location of the joints at each frame. We also provide results after applying temporal smoothing to the predictions via the Viterbi algorithm where the transition function is the Euclidean distance of the same joints in two neighboring frames. Additionally, we show the performance of a convolutional RNN with  $w_{i,t}^y = 0, \forall i, t$  in Eq. 4. This model corresponds to a standard convolutional RNN where the output variables of the previous time steps are not connected to the hidden state. All networks have roughly the same numbers of parameters, to ensure a fair comparison. For our chain model in time, we show results for two choices of time horizon  $T_H$ . Namely,  $T_H = 1$ , where predictions of only the previous time step are being considered and  $T_H = 3$ , where predictions of the



**Fig. 6.** Examples of predictions on the Penn Action dataset. Predictions by the per frame model (top) and by the chain model (bottom) are shown in each example block.

past 3 frames are considered at each time step. Finally, we show the performance of a chain model in time and in joints, with a time horizon of  $T_H = 3$ .

We compare to previous work on the Penn Action dataset [42]. This model uses action specific pose models, with shallow hand-crafted features, and improves upon Yang and Ramanan [32].

We observe a gain in performance compared to the per frame CNN as well as the RNN across all joints. Interestingly, chain models show bigger improvement for arms compared to legs. This is due to the fact that the people in the videos play sports which involve big arm movements, while the legs are mostly unoccluded and less kinematic. In addition, we see that  $T_H = 3$  leads to better performance, which is not surprising since the model makes a decision about the location of the joints at the current time step based on observation from 3 past frames. We did not observe additional gains for  $T_H > 3$ . Chaining in time and in joints does not improve performance even further, possibly due to the already high accuracy achieved by the chain model in time.

Figure 6 shows examples of predictions by our chain model on the Penn Action dataset. We also show the predictions made by the per frame detector. We see that the chain model is able to disambiguate right-left confusions which occur often due to the constant motion of the person while performing actions, while the per frame detector switches very often between erroneous detections.

## 5 Conclusions

In this paper, motivated by sequence-to-sequence models, we show how chained predictions can lead to a powerful tool for structured vision tasks. Chain models allow us to sidestep any assumptions about the joint distribution of the output variables, other than the capacity of a neural network to model conditional distributions. We prove this point experimentally by showing top performing results on the task of pose estimation from images and videos.

## References

1. Daumé Iii, H., Langford, J., Marcu, D.: Search-based structured prediction. *Mach. Learn.* **75**(3), 297–325 (2009)
2. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: *NIPS* (2014)
3. Chan, W., Jaitly, N., Le, Q.V., Vinyals, O.: Listen, attend and spell. *arXiv preprint arXiv:1508.01211* (2015)
4. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164 (2015)
5. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1538–1546 (2015)
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440 (2015)
7. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: *CVPR* (2015)
8. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.0473* (2014)
9. Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., Hinton, G.: Grammar as a foreign language. In: *Advances in Neural Information Processing Systems*, pp. 2755–2763 (2015)
10. Vinyals, O., Bengio, S., Kudlur, M.: Order Matters: sequence to sequence for sets. *ArXiv e-prints*, November 2015
11. Goldberg, Y., Elhadad, M.: An efficient algorithm for easy-first non-directional dependency parsing. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 742–750 (2010)
12. Ross, S., Gordon, G.J., Bagnell, J.A.: A reduction of imitation learning and structured prediction to no-regret online learning. *ArXiv e-prints*, November 2010
13. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. *Int. J. Comput. Vis.* **61**(1), 55–79 (2005)
14. Ramanan, D.: Learning to parse images of articulated bodies. In: *NIPS* (2006)
15. Andriluka, M., Roth, S., Schiele, B.: Pictorial structures revisited: people detection and articulated pose estimation. In: *CVPR* (2009)
16. Eichner, M., Ferrari, V.: Better appearance models for pictorial structures (2009)
17. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: *CVPR* (2011)
18. Johnson, S., Everingham, M.: Learning effective human pose estimation from inaccurate annotation. In: *CVPR* (2011)
19. Tian, Y., Zitnick, C.L., Narasimhan, S.G.: Exploring the spatial hierarchy of mixture models for human pose estimation. In: *Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS*, vol. 7576, pp. 256–269. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33715-4\_19
20. Wang, F., Li, Y.: Beyond physical connections: tree models in human pose estimation. In: *CVPR* (2013)
21. Sapp, B., Taskar, B.: Modec: multimodal decomposable models for human pose estimation. In: *CVPR* (2013)

22. Pishchulin, L., Andriluka, M., Gehler, P., Schiele, B.: Poselet conditioned pictorial structures. In: CVPR (2013)
23. Karlinsky, L., Dinerstein, M., Harari, D., Ullman, S.: The chains model for detecting parts by their context. In: CVPR (2010)
24. Toshev, A., Szegedy, C.: Deeppose: human pose estimation via deep neural networks. In: CVPR (2014)
25. Ramakrishna, V., Munoz, D., Hebert, M., Andrew Bagnell, J., Sheikh, Y.: Pose machines: articulated pose estimation via inference machines. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part II. LNCS, vol. 8690, pp. 33–47. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-10605-2\\_3](https://doi.org/10.1007/978-3-319-10605-2_3)
26. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback (2015)
27. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: CVPR (2016)
28. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, attend and tell: neural image caption generation with visual attention. CoRR abs/1502.03044 (2015)
29. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: NIPS (2015)
30. Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B.: 2D human pose estimation: new benchmark and state of the art analysis. In: CVPR (2014)
31. Zhang, W., Zhu, M., Derpanis, K.: From actemes to action: a strongly-supervised representation for detailed action understanding. In: ICCV (2013)
32. Yang, Y., Ramanan, D.: Articulated human detection with flexible mixtures-of-parts. PAMI (2012)
33. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814 (2010)
34. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint [arXiv:1502.03167](https://arxiv.org/abs/1502.03167) (2015)
35. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR abs/1512.00567 (2015)
36. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 340–353. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-33712-3\\_25](https://doi.org/10.1007/978-3-642-33712-3_25)
37. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. CoRR abs/1603.06937 (2016)
38. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: CVPR (2015)
39. Hu, P., Ramanan, D.: Bottom-up and top-down reasoning with hierarchical rectified gaussians. CVPR (2016)
40. Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P., Schiele, B.: Deeppcut: joint subset partition and labeling for multi person pose estimation. CVPR (2016)
41. Lifshitz, I., Fetaya, E., Ullman, S.: Human pose estimation using deep consensusvoting. CoRR abs/1603.08212 (2016)
42. Xiaohan Nie, B., Xiong, C., Zhu, S.C.: Joint action recognition and pose estimation from video. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015