Procedural Activity Recognition and Mistake Detection

Filippos Bellos
fbellos@umich.edu

Tom Brady
tdbrady@umich.edu

Lydia Rogers
lydiajr@umich.edu

Abstract

We consider two highly codependent problems. First, fine-grained, multi-step activity recognition from instructional videos containing different recipes spanning up to several minutes. Second, we examine step mistake recognition while executing recipe instructional videos. The first problem we address differs from traditional activity classification due to the length of the videos. Typical models are trained on short videos spanning only a few seconds. These videos are manually trimmed to contain simple atomic actions. Our approach leverages robust activity recognizers using Contrastive Captioners (CoCa) and distant supervision learning methods, optimizes the models on downstream tasks, and creates a new "mistake" dataset. We compare these two approaches and present a model capable of recognizing mistakes with high accuracy.

1 Introduction

We address a multi-faceted problem. First, finding the best manner in which to extract meaningful embeddings from instructional videos that last several minutes. We explore multi-modal image-text representations pretrained on a large dataset, expecting this to yield the best results.

The second problem is leveraging these learned rich embeddings of each step's sematics to classify temporal segments of steps provided in recipe instructions (**multi-step activity recognition**). Additionally, we investigate how we can temporally localize when an out of order mistake has been executed (**mistake detection**).

2 Related Work

This section details previous work that relates to our recipe mistake detection task.

2.1 Written Recipe Recognition

Computer recognition and understanding of recipe instructions has been heavily researched within NLP over the last decade. Donatelli et al. (Donatelli et al., 2021) translated recipes into a graph structure in hopes of recognizing different recipes for the same dish as equivalent steps in preparing that dish. By aligning recipes at the action level, the complex structure of these recipes could be modeled. For example, Figure 1 shows two different recipes for making waffles, with "preheat" in (b) clearly aligned with "preheated" in (c), giving useful insight into recipe step recognition. The graphing method proved effective at aligning recipes, and the novel corpus that was created to model alignment across different recipe steps provides a useful basis for training models in recipe step recognition.

(b) **Preheat** your waffle iron. In a large bowl, **mix** together the flour, salt, baking powder, and sugar. In another bowl, **beat** the eggs. **Add** the milk, butter, and vanilla to the eggs. **Pour** the liquid into the flour mixture and **beat** until blended. **Ladle** the batter into the waffle iron and **cook** until crisp and golden.

(c) **Sift** together in a large mixing bowl flour, baking powder, salt, and sugar. In a jug, **measure** out milk. **Separate** eggs, **placing** egg whites in the bowl of standing mixer. **Add** yolks and vanilla essence to milk and whisk together. **Pour** over the flour mixture and very gently **stir** until **combined**. **Stir** in the **melted** butter and **continue mixing** very gently until combined. **Beat** egg whites until stiff and slowly **fold** into batter. **Spoon** the batter into **preheated** waffle iron in batches and **cook** according to its directions. **Remove** immediately and **serve** with maple syrup and fruits.

Figure 1: Two different recipes for making waffles (Donatelli et al., 2021). The "preheat" in (b) aligns with "preheated" in (c), giving meaningful relations between the two in recognizing them as equivalent steps in making waffles.

2.2 Recipe Image Recognition

While written recipe recognition is a key step in recipe understanding, computer vision plays a significant role as well. Research into computer vision methods has allowed recipe preparation and instruc-

tion videos to be connected to written recipes. This has enabled recipe video annotating, i.e. adding annotations for recipe videos with their accompanying steps. To that end, multimodal pre-training techniques have been developed. Min et al. (Min et al., 2017) introduced multimodal content modeling, which is utilized to accomplish tasks like identifying ingredients and attributes of recipes from images. They proposed the MultiModal MultiTask Deep Belief Network (M^3TDBN). As shown in Figure 2, the first pathway learns the join representation of image features and visible ingredients. The second pathway learns the representation of ingredients, which includes non-visible ingredients. At the top layers, this information is connected, enabling fine-tuning of the whole architecture as shown in Figure 2, which can classify the cuisine and course in a downstream task.

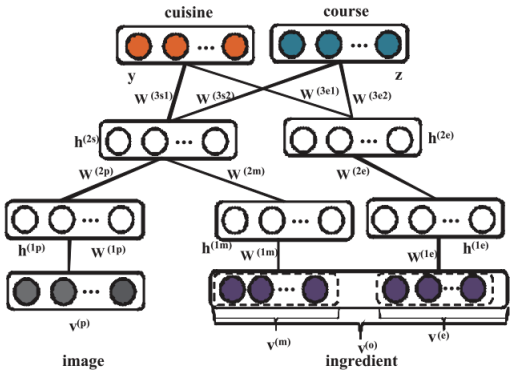

Figure 2: A multimodal model with joint pathways for identifying visible ingredients and non-visible ingredients (Min et al., 2017). This model enables fine-tuning such as classifying course and cuisine from images and ingredients.

These techniques and others have greatly advanced processing recipe text and videos using natural language processing and computer vision techniques. However, there is not much research in the area of error detection in recipe recognition. Our approach is among the novel approaches in this area, which combines prior techniques in image-text pre-training and the resulting models to accomplish the task.

2.3 Image-Text Pretraining

Recent work has proposed image-text foundation models that can subsume both vision and vision-language pretraining. CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021) proved that when dual-encoder models are pretrained on noisy image-text pairs using contrastive learning can learn strong image and text representations for cross-modal alignment tasks. CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which text in the dataset (OpenAI, 2021). CLIP can then be used as a zero-shot classifier, predicting captions for images it has not yet seen. It is highly efficient, flexible, and general, pairing image and text as we desire. ALIGN implements a dual-encoder as well, pushing embeddings of matched image-text pairs together and non-matched pairs apart (Jia et al., 2021). These two methods prove useful when performing cross-modal alignment tasks, like our recipe step recognition. A different angle proposes encoder-decoder models trained with generative loss and achieves strong performance on vision-language tasks (Wang et al., 2021), (Wang et al., 2022). Simple Visual Language Model Pre-training with Weak Supervision (SimVLM) trains on weakly aligned image-text pairs, i.e. text paired with an image where the text is not necessarily a precise description for the image. This encoder-decoder model achieves strong performance as well, performing well in zero-shot image classification.

2.4 Distant Supervision and Contrastive Captioners

Distant supervision has been studied in natural language processing and usually the supervision for the training is obtained by automatically mining examples from a large noisy corpus utilizing a clean and informative knowledge base (Alayrac et al., 2020). It has been shown to be very successful on the problem of relation extraction. However, the concept of distant supervision had not been exploited in video understanding prior to (Lin et al., 2022).

Contrastive Captioners (CoCa) has been successful in creating aligned unimodal image and text embeddings. We leverage these two models in achieving recipe step mistake detection.

3 Datasets

This section details the datasets that we use in pre-training the models and in downstream tasks.

3.1 COCO - Common Objects in Context

COCO (Lin et al., 2014) is a dataset of 330,000 images, with over 200,000 containing associated labeled captions. There are 91 classes of images,

with 80 in use. Each labeled image has five associated captions. This dataset has been utilized widely since its creation in 2015 to train image captioning and recognition models. We utilize this dataset to pre-train our CoCa model. This dataset is ideal for training with CoCa due to its focus on object identification. Since CoCa heavily relies on learned tensors in order to properly identify objects, COCO's focus on object segmentation makes it an ideal dataset for pretraining this model.

3.2 HowTo100M

HowTo100M (Miech et al., 2019) is a dataset of embeddings of over 1M long instructional videos split into about 120M video clips in total. Importantly, HowTo100M has auto-generated captions, creating noisy image-text pairs. This dataset is used for the distant supervision method that essentially denoises the ASR textual descriptions. However, due to the size and formatting of this particular dataset, we utilize the previously mentioned COCO dataset rather than HowTo100M for CoCa pretraining.

3.3 COmprehensive INstructional Video Analysis (COIN)

The dataset utilized for downstream tasks is COIN, a large-scale dataset for comprehensive instructional video analysis (Yansong Tang, 2019). This dataset offers a wide variety of encoded long-form videos, where each instruction step shown in the video is encoded along with the video timestamp boundaries for each step. Specifically, for our downstream tasks, we utilize the "boil noodles" subset of COIN, which contains 97 instructional videos of how to boil noodles with up to four steps each. This subset contains several instructional videos for the task of boiling noodles, with their associated steps. We have chosen this particular dataset for both its simplicity and size. Boiling noodles is a common instructional procedure, and across different instructional videos, the primary objects for our pretraining methods to recognize, such as water, a pot, and the noodles themselves, appear in each video. When segmenting each of the 97 videos into frames for modeling, we have very sufficient training data for our desired results.

In order to create our "mistake" dataset, we generate a process to create lapse (out of order) mistakes in the instructions associated with the boiling noodles subset. The recipes are executed in a shuffled order as shown in Figure 3. Our goal is for our approaches to be able to utilize predictions of the next frame of the video, as well as learned features of the video, to be able to identify with confidence when the actions in the video do not align with the recipe instructions.

4 Distant Supervision

The goal of this method is to learn a segment-level representation to express a long procedural video as a sequence of step embeddings. Then a sequence model can be applied on this video representation to perform temporal reasoning over the individual steps. While step annotations could enable the training of models to recognize the individual steps of procedural activities, existing large-scale datasets in this area do not include such segment labels due to the prohibitive cost of manually annotating temporal boundaries in long videos. That is why we want to learn the step-level representation without manual annotations, so as to enable training on these large-scale unlabeled data.

4.1 Leveraging Knowledge Base to denoise ASR labels

WikiHow acts as a knowledge base B containing textual step descriptions for T tasks: $B = \{y_1^{(1)}, ..., y_{S_1}^{(1)}, \ldots, y_1^{(T)}, ..., y_{S_T}^{(T)}\}$, where $y_s^{(t)}$ represents the language-based description of step s for task t, and S_t is the number of steps involved for the execution of task t. An instructional video x is expressed as a sequence of L segments $\{x_1, ..., x_l, .., x_L\}$, with each segment x_l consisting of F RGB frames. Each video is accompanied by a paired sequence of text sentences $\{a_1, ..., a_l, .., a_L\}$ obtained by applying ASR to the audio narration. This narration a_l is noisy due to ASR errors.

So, by using this distant supervision method we can leverage the knowledge base B to denoise the narration a_l and to convert it into a supervisory signal that is more directly related to the steps represented in segments of the video. In order to be able do that without any form of annotation we approximate the unknown conditional distribution:

$$P(y_s^{(t)}|x_l) \approx \frac{\exp\left(\mathcal{S}(a_l, y_s^{(t)})\right)}{\sum_{t,s} \exp\left(\mathcal{S}(a_l, y_s^{(t)})\right)}. \quad (1)$$

over the steps executed in the video after first calculating the textual similarity measure $\mathcal{S}$ between $y_s^{(t)}$ and a_l :

$$\mathcal{S}(a_l, y_s^{(t)}) = e(a_l)^\top \cdot e(y_s^{(t)}) \quad (2)$$

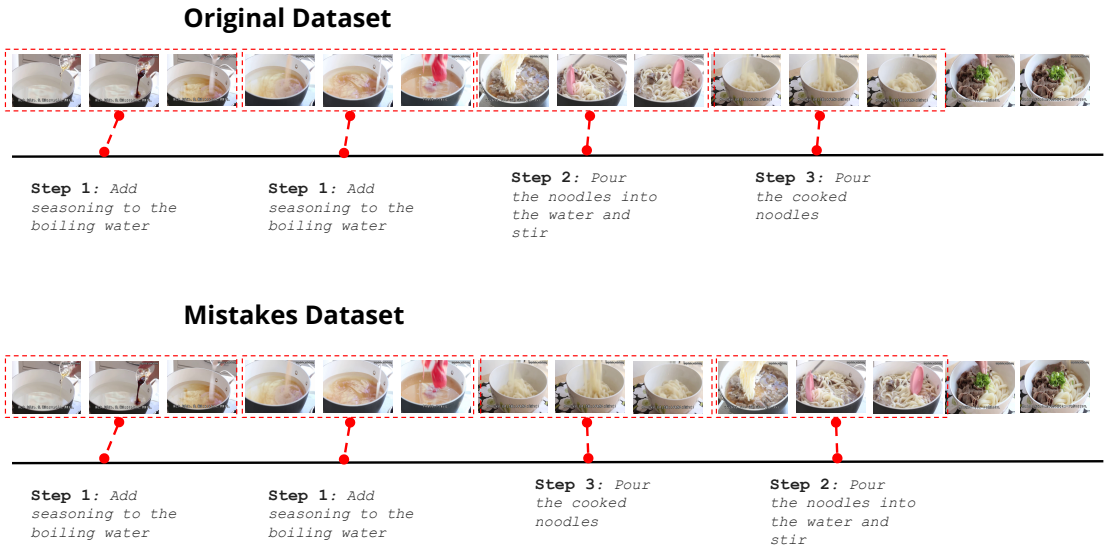

Figure 3: Video example from mistake dataset

where $e(a_l), e(y_s^{(t)}) \in R^d$ and d is the dimension of the language embedding space.

So, we use that approximated distribution $P(y_s^{(t)}|x_l)$ as the supervision to learn the video segments representation, since it is more salient compared to the noisy and unstructured narration a_l. The training objectives to learn that video representation are explained below.

4.2 Learning meaningful representations from videos

In this work, three different training objectives are used to train a Timesformer to recognize individual steps of procedural activities in video.: (1) step classification, (2) distribution matching, and (3) step regression. For (1): the target indices t^*, s^* of the steps in the Knowledge Base that best describes a given video segment is basically the argmax of the target conditional distribution

$$t^*, s^* = \arg\max_{t,s} P(y_s^{(t)}|x_l). \tag{3}$$

So with these indices as targets we train a classification model to learn embeddings using a standard cross-entropy loss:

$$\min_{\theta} -\log\left([\mathcal{F}_C(x_l;\theta)]_{(t^*,s^*)}\right) \tag{4}$$

where $\mathcal{F}_C$ denotes the activity recognition model on step level and θ denotes the learning parameters of that model.

For (2): we train the step classification model to minimize the KL-Divergence between the predicted distribution and the target distribution, which is the conditional probability we have been talking about:

$$\min_{\theta} \sum_{t,s} P(y_s^{(t)}|x_l) \log \frac{P(y_s^{(t)}|x_l)}{[\mathcal{F}_C(x_l;\theta)]_{(t,s)}}. \tag{5}$$

For (3): we train the video model to predict the language embedding $e(y_{s^*}^{(t^*)})$ associated to the pseudo ground-truth step, which is the same indices from (1), t^*, s^* . So, again from argmax of the conditional distribution.

For the training, the NCE loss is used as the objective:

$$\min_{\theta} -\log \frac{\exp\left(e(y_{s^*}^{(t^*)})^\top \mathcal{F}_R(x_l;\theta)\right)}{\sum_{(t,s)\neq(t^*,s^*)} \exp\left(e(y_s^{(t)})^\top \mathcal{F}_R(x_l;\theta)\right)} \tag{6}$$

where $\mathcal{F}_R$ denotes the regression model that does the mapping to the language embedding space.

4.3 Downstream Tasks

4.3.1 Activity Recognition

We make use of $\mathcal{F}_C(x_l)$ explained in 4.2 as a feature extractor to capture step-level information from new video segments. Specifically, we use the second-to-last layer of $\mathcal{F}_C(x_l)$ (before the softmax function) as the step embedding representation

$f(x_l)$ for classification of procedural activities in long videos. We then train a linear classifier on top of those step embedding representations, using our new dataset.

4.3.2 Activity Forecasting

We consider the task of "next-step anticipation". To do that, the proposed classification model is modified to address forecasting tasks over a sequence of steps to predict future activity.

So, given as input a video spanning M segments, $\{x_1, ..., x_M\}$, the goal is to predict the step executed in the unobserved $(M+1)$ segment. In order to do that a transformer is trained. So, for each input segment x'_l, we include $f(x'_l)$ which is the embedding of the best matching step $\hat{y}(x_l) = y_{s'}^{t'}$ but also $e(y_{s'+1}^{t'})$, which is the embedding of the step immediately after the step matched in the knowledge base. This effectively provides the transformer with information about the likely future steps according to the knowledge base.

4.3.3 Mistake Recognition

We are using the activity recognition model described in 4.3.1 as our initial state estimation predictor. So, the output of that model is converted to normalized probabilities (0 to 1) corresponding to each possible state. The maximum of those predictions is selected, applying a threshold on it, as well as the state it corresponds to. The threshold makes sure that the activity recognition model is confident enough for its prediction. This confident output is added to a fixed length window. Whenever the window is full we calculate the most frequent estimated state that populates the window(most occurrences), and then we empty the window. This, most frequent, estimated state is then given as input to the orchestration model which will decide if a mistake has occurred. A mistake occurs if one of the model's constraints is violated. These constraints aim to preserve the sequential characteristics of the recipe being executed and are explained below. Given the steps in order from the original dataset labels:

- The initial state has to be state 0 or the first state in the labels if state 0 is omitted, and the ending state has to be state 3 or the last state in the labels . For example, if the first model's input is a state corresponding to a state other than 0 or first state then a mistake has occurred.
- The only actions allowed are the one that stays at the same state, and the one that transitions to the next state. So for example, if the model receives as input, state 3 or state 1 while the previous received state was state 2, then none of the two aforementioned actions is performed and a mistake is detected.

5 Contrastive Captioners (CoCa)

This section details our second approach, which leverages Contrastive Captioners (CoCa) for pre-training.

5.1 CoCa Model

The CoCa model (Yu et al., 2022) is a novel encoder-decoder strategy that generates aligned unimodal image and text embeddings and joint multimodal representations, making it versatile enough to be used for a variety of downstream applications. CoCa, in particular, produces cutting-edge performance on vision and vision-language tasks, including vision identification, cross-modal alignment, and multimodal understanding. It also learns extremely generic representations, allowing it to compete with or outperform thoroughly fine-tuned models using zero-shot learning or frozen encoders.

CoCa is a unified training system that effectively combines single-encoder, dual-encoder, and encoder-decoder paradigms by integrating contrastive loss and captioning loss on a single training data stream consisting of image level annotations and noisy image-text pairs.

A new encoder-decoder architecture is introduced in CoCa. The encoder is a vision transformer (ViT) (Dosovitskiy et al., 2020), and the text decoder transformer is split into two parts: a unimodal and a multimodal text decoder. Multimodal decoder layers are cascaded with cross-attention to image encoder outputs to learn multimodal image-text terms for captioning loss. This approach maximizes the model's versatility and universality in accommodating a wide range of jobs while also being trained effectively with only one forward and backward propagation for both training objectives, resulting in little computational overhead with an ideal implementation. The architecture of CoCa is shown in Figure 5, which takes image-caption pairs, running images through the ViT image encoder and text decoder, which with an attentional pooler can be fed to a multimodal text decoder to generate captioning loss.

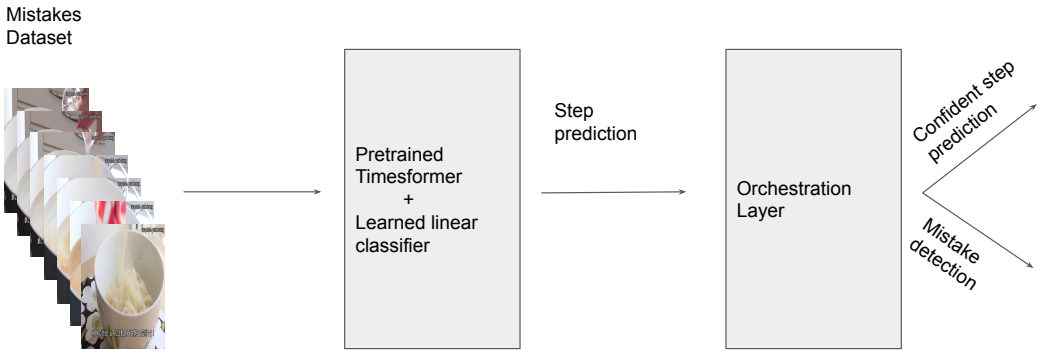

Figure 4: Overview of mistake recognition module

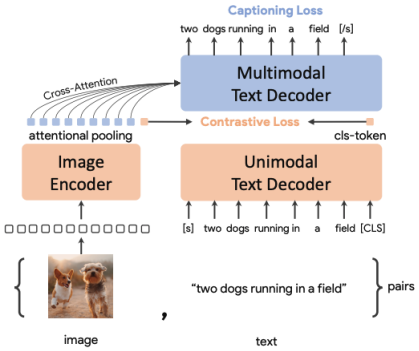

Figure 5: CoCa architecture (Yu et al., 2022). The CoCa model takes image-text pairs in the pre-training task, generating captioning loss in training the model.

As a result, the model can be trained from the ground up with costs comparable to a naive encoder-decoder model. We use a simple approach to enable a learned CoCa model for video activity recognition tasks. We first take multiple frames of a video and feed each frame into the shared image encoder individually. For frozen feature evaluation or finetuning, we learn an additional pooler on top of the spatial and temporal feature tokens with a softmax cross-entropy loss. Note the pooler has a single query token thus the computation of pooling over all spatial and temporal tokens is not expensive.

5.2 CoCa Pre-Training

The code base for Google AI's implementation for CoCa is not publicly available. In order to proceed with our analysis, we utilize an open-source implementation of CoCa on Github (Wang, 2022). This code utilizes PyTorch and ViT, allowing us to implement CoCa in our own work using open-source packages.

To accompany the open-source implementation, we build a process to load images from COCO (Lin et al., 2014) for the purpose of pretraining the model. Using the captioned subset of COCO, we load images from their URLs stored in a COCO JSON file, tokenizing the caption and converting the image into a 256 x 256 tensor. These image text-token pairs are then passed to CoCa, which trains on thirty-two example-batches over several epochs, computing the captioning loss to learn the image-text relations. This training involves computing the contrastive and captioning losses, the first of which is produced through the attentional pooling process and the latter computed following the use of a multimodal text decoder for captioning.

5.3 Downstream Tasks

5.3.1 Activity Classification

Once pretraining with CoCa is completed, we will then use the feature representation that we learned and the COIN dataset to train a simple linear classifier. The linear classifier works by utilizing key features of each input video frame provided by CoCa

and estimating linear boundaries based on these features to minimize classification error. Through this procedure, we are able to identify which sections of our recipe videos from COIN align with which instructions in the recipe instructions given. The goal of this classifier is to correctly identify these video segment-recipe step pairings, i.e. state estimation on the frame level, which will prove useful in state-prediction for later mistake detection tasks.

5.3.2 Video Recipe Activity Recognition

We additionally fine-tune our CoCa model to perform video recipe step recognition. The procedure for video interpretation using CoCa is shown in Figure 6 below. After extracting the frames of a subset of each stepwise boundary of a boiling noodles video, we pass each into the CoCa image encoder, utilizing an attentional pooler to generate output embeddings. This attentional pooler identifies key components in the collection of video frames, as well as the locations and state changes associated with each object. We can then perform softmax cross-entropy loss to perform activity forecasting. In this, we predict the $(n+1)$th frame with the first n frames given as input to the task. To perform mistake detection, we need this method to understand the correct temporal relations of different frames.

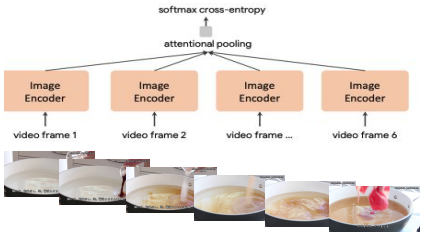

Figure 6: CoCa for video representation (Yu et al., 2022) using individual image encoders, attentional pooling, and softmax cross-entropy loss.

5.3.3 Mistake Recognition

Similar to distant supervision, we use our state-prediction model as our initial state estimation predictor. The output is then converted to normalized probabilities, corresponding to each possible state. We select the maximum of those predictions and apply a threshold on it, ensuring that the model is confident enough for its prediction. We then detect mistakes by seeing if one of the model's constraints is violated, which are outlined in 4.3.3.

5.4 Limitations of CoCa

While the CoCa architecture can be incredibly powerful and accurate when provided the proper resources, the time and data required to achieve desirable results from CoCa is immense. For example, using our batch size of 32 images per pretraining iteration, a single batch of images requires over a minute to compute the loss on the Linux server utilized. This is likely due to the intense process of computing contrastive loss in order to locate common tensor patterns between images. While the captioning loss is quite straightforward to compute, the time complexity of contrastive loss increases immensely with each additional image. This induces a tradeoff between achieving higher accuracy on downstream tasks and the time and resources available for pretraining. For this reason, we use only a subset of images from COCO to pretrain the model. This tradeoff, as well as the fact that a pretrained CoCa model and the pretraining data from the original report are not publicly available, limits the success we can achieve with this model.

6 Method Evaluation

Our main goal was to compare the two approaches and attempt recipe mistake detection, given the embeddings extracted by them. The process of learning these embedding with these methods differs significantly. The distant supervision method automatically identifies steps in instructional videos by leveraging the distant supervision of a textual knowledge base (wikiHow) that includes detailed descriptions of the steps needed for the execution of a wide variety of complex activities. So, it basically tackles the problem of existing large-scale datasets in this area not including such segment labels due to the prohibitive cost of manually annotating temporal boundaries in long videos. Then, these denoised textual descriptions are used to learn video segment representations by training a Timesformer model. As for the second method, CoCa is a minimalist design that pretrains an image-text encoder-decoder foundation model jointly with contrastive loss and captioning loss, thereby combining model capabilities from contrastive approaches like CLIP and generative methods like SimVLM. So, it follows the trend of using large-scale pretrained foundation vision-language models.

We wanted to compare these modern approachs to conclude which one yields the richer representations, in order to later use them for our downstream

Task	Segment Model	Pretraining Dataset	Eval. Dataset	Acc(%)
Activity Recognition	Timesformer	HT100M	COIN Subset	70.6
Activity Forecasting	Timesformer	HT100M	COIN Subset	51.2
Mistake Detection	Timesformer	HT100M	Mistake COIN Subset	63.5

Table 1: Results with the distant supervision model on activity recognition, activity forecasting, and mistake detection.

tasks and eventually the novel task of mistake detection.

7 Results

The results we obtained from distant supervision were quite encouraging. As shown in Table 1.

We can see the effectiveness of the step embedding extracted from the denoising process, as with just a trainable classifier and a transformer in the case of Activity Recognition and Activity Forecasting, we achieve a decent performance that translates into a decent mistake recognition as well. The distant supervision model lends itself to expressing long videos as a sequence of step embeddings, proving to be effective when fine-tuned for downstream tasks.

CoCa proved to be a far more difficult approach than distant supervision. While the model was pretrained with limited data as explained above, the generated embeddings were both insufficient and highly-complex due to the intricacies of CoCa. Although we did manage to successfully implement the downstream tasks, due to not having the resources to pre-train CoCa on an adequate amount of data, the results obtained were not sufficient to report.

8 Experiments

Due to the computational cost, for all the downstream tasks, we chose to use the already extracted text embeddings of all the ASR sentences in HowTo100M provided by the authors as well as the pretrained weights for the video model, Timesformer. The dataset used for the pretraining is HowTo100M (HT100M) as explained in section 3. To evaluate our downstream tasks, we use the step annotations from COIN. We chose to use only one of the procedural activities in order to be more flexible and be able to evaluate on all our downstream tasks. This recipe is number 10 in the COIN dataset and called BoilNoodles. It consists of 97 videos, where each one is comprised of up to 4 steps as explained in Section 3. The steps are manually annotated within each video with temporal boundaries and step class labels.

The Timesformer used starts from a configuration of ViT initialized with ImageNet-21K ViT pretraining. Each segment consists of 8 frames uniformly sampled from a time-span of 8 seconds. For pretraining, the segments are sampled according to the ASR temporal boundaries available in HowTo100M while a batch of 256 segments is used.

For activity recognition, we train a simple linear classifier on embeddings extracted from individual segments of the COIN dataset. We used sampling of 8 frames uniformly, as well as experimented with less frames. That didn't have much difference on the performance.

For activity forecasting on the COIN subset we use a single basic transformer layer trained on top of our fixed embeddings. The transformer layer that performed best has 768 embedding dimensions and 12 heads, while we also tried one with 512 embedding size and 8 heads. The difference in performance was about 4%.

For the task of mistake recognition we experimented with different thresholds of prediction confidence, ending up using a threshold of 80%. We also tried different window sizes. As explained in section 4.3.3, the window is used to compute the most frequent predicted activity in it. And since we used sampling of 8 frames for each video segment the window sizes we tried varied from 4 to 8. We ended up using 8 but the difference in performance was minimal, around 1% between the worst window size and the best.

9 Conclusion

For this project, we investigate the use of two pretraining methods for image recognition and captioning, and use the results in performing a novel downstream task of mistake recognition. Although we do not succeed in performing this downstream task using the CoCa model we trained due to time and resource restraints, the use of distant super-

vision offers promising results, particularly with activity recognition and mistake detection. This is an encouraging result for future work, and hints at how these methods could be utilized in actively recognizing the task that a user is trying to accomplish and guides him step-by-step in the successful execution of the recipe, detecting when a mistake has occurred. Mistake detection for instructional procedures is shown to be an achievable and worthy milestone in the application of computer vision and natural language processing.

References

Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. 2020. Self-supervised multimodal versatile networks.

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. 2021. Aligning actions across recipe graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6930–6942.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft coco: Common objects in context.

Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and Lorenzo Torresani. 2022. Learning to recognize procedural activities with distant supervision.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips.

Weiqing Min, Shuqiang Jiang, Jitao Sang, Huayang Wang, Xinda Liu, and Luis Herranz. 2017. Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration. *IEEE Transactions on Multimedia*, 19(5):1100–1113.

OpenAI. 2021. Clip: Connecting text and images.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework.

Phil Wang. 2022. Coca - pytorch.

Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. 2021. Simvlm: Simple visual language model pretraining with weak supervision.

Yongming Rao Yu Zheng Danyang Zhang Lili Zhao Jiwen Lu Jie Zhou Yansong Tang, Dajun Ding. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. 2022. Coca: Contrastive captioners are image-text foundation models.