Thesis for the degree
Master of Science

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

By
Shahaf Wagner

עבודת גמר (תזה) לתואר
מוסמך למדעים

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

מאת
שחף וגנר

למידה וביצוע של מספר תפקידים בו-זמנית לעומת אחד אחרי השני
Learning and executing multiple tasks together vs. one task at a time

Advisor:
Professor Shimon Ullman

מנחה:
פרופסור שמעון אולמן

November 2020

חשוון ה'תשפ"א

Brief summary

The analysis of a complex scene requires the application of a considerable number of visual tasks, well beyond what is performed in current models. Multiple objects in the scene need to be recognized and located, together with their properties and inter-object relations. A single scene may contain a large number of objects, and objects parts, together with their properties and relations, and the total number of object classes, properties and relations for a human-level scheme is estimated to be in the tens of thousands[1]. It is consequently infeasible to learn all the tasks simultaneously or to extract the full structure of complex scenes. As discussed below, problems of combinatorial generalization further prohibit the simultaneous learning and execution of multiple tasks simultaneously. We conclude that it is often desirable to learn one task at a time. On the other hand, during scene analysis, to obtain a fast response it is desirable to perform multiple tasks simultaneously whenever possible. We propose an approach in which the model can be instructed to learn and to execute specific visual tasks, either individually or in combinations. We show how it is possible in this model to learn tasks individually, but then perform multiple tasks together without any additional learning. The ability to perform multiple tasks within a single step is subject to limitations, which are demonstrated and analyzed in our work.

Using the 'Avatars' data set of synthetic persons with associated properties, we learn the individual tasks of extracting a selected property of a selected person. We then test the extraction of multiple properties of multiple persons, without additional training. We find that we can extract multiple properties of a selected person without compromising accuracy. Extracting a given property of different persons is shown to be limited by the 'binding problem': the model extracts the correct property value for the two persons, but cannot associate correctly the persons with the correct property values. We further show that the simultaneous extraction of multiple properties deals well with combinatorial generalization, to combinations of person-property not seen in training. Using an array of MNIST digits, we learn to extract spatial relations (left, right, above, below), and a single task to perform is to extract, e.g., 'the digit above the 5'. We show that the ability to combine individual tasks is limited again by the binding problem. If we use a common representation for the output digits (a single representation of 10 digits for all 4 relations), binding becomes ambiguous. If we use separate representation (e.g., '5' as a left-neighbor is different from a right-neighbor), then executing multiple relations simultaneously becomes possible. We finally show that the use of separate representations comes however with a price of reduced combinatorial generalization, while the common representation does not suffer from the issue.

Contents

1 Introduction

1.1 Background and Motivation

Analyzing a scene typically requires applying multiple visual tasks to the same image, such as classifying several objects, segmenting some of them, identifying their parts, and extracting multiple properties and relations. Therefore, a general problem is how to deal with the learning and then the execution of multiple tasks. Should we learn and execute numerous tasks together, such as classifying and segmenting all the objects in the image, applying classification and segmentation to one object at a time, or using some intermediate approach? A problem arises in both learning and execution. We could, for example, learn to classify one object at a time but then recognize all the objects in the image together. Alternatively, we could learn to identify all the properties of a given object together but then extract them when required, one at a time.

From a practical consideration, it is often desirable to obtain relevant information from the scene as rapidly as possible. It is therefore desirable to perform a large number of visual tasks simultaneously. Currently, this is the dominant approach in computer vision applications, using either the extraction and processing of a large number of region proposals across the image as in RCNN Ross et al.[2](2014), fast-RCNN[3], and mask-RCNN[4], or the processing of multiple image regions arranged on a regular grid such as Yolo[5] and YoloV3[6]. There are, however, two limitations to the simultaneous application of a large number of tasks. The first and obvious restriction comes from capacity limitations. A fixed network is expected to have a limited computational capacity, and will therefore, be limited in the tasks it can support simultaneously. Recent work has shown that multiple tasks, even just a few, can reduce the performance of each of them when the tasks are performed together in the same network as in[7], which discusses multi-task learning as a problem of multi-objective optimization.

A second limitation, described in more detail below, is that dealing with multiple tasks simultaneously has an adverse effect on the ability of the network to generalize. In addition, the problem of learning and execution needs to be addressed separately. During learning, time considerations are often less important than execution time, but the complexity of the learning can play an important role. Learning multiple tasks together could be a complex learning problem, requiring, for example, a large number of training images, and therefore different schedules may be applied in learning and in execution. In the current work, we will examine a number of issues in learning and use of multiple tasks vs. one task at a time.
Based on past results, our starting point is that learning multiple tasks together is often impractical due to the complexity of the learning task and the lack of combinatorial generalization. We focus on the problem of learning one task at a time but later performing multiple tasks together.

1.2 Related Work

There is almost no literature on the problem of learning and executing one task vs. multiple tasks, except for the work in [8] , [9] , [10] , [11] and [12] which talk about task selection, but not the issue of selecting multiple tasks. Previous work usually dealt with a single task, such as recognizing a single object in ImageNet. More recent work has dealt with doing several tasks together, such as Mask-RCNN[4] and YOLO[5], which perform recognition, segmentation, and human keypoint detection, for all objects in the image. In these approaches, all the tasks are learned together and executed simultaneously, with no control over which task, or combination of tasks, to perform.

Related literature comes from papers using attention to guide the computation. The task to perform is usually selected by selecting a particular sub-region of the image, where the processing is being applied. For example, in the work on 'Learn to Pay Attention' [13] the authors used an attention mechanism to focus on key regions of the image that are important for the task. They demonstrated improvement in performance on multi-class classification, using attention estimators to calculate a mask for the relevant parts in the image. Using this technique, they obtained 7% improvement in performance compared with a baseline. Anohter paper that uses sequential attention is 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention' [14] . The task in this case was image captioning, in which a CNN processes the image and then an RNN generates a description of the image. For each word in the description, the model focuses on a different part of the image, and in this manner the overall task is obtained by a sequence of sub-tasks. Unlike the current work, the tasks in the attention models are selected only be selecting a location to apply the processing to, and there is not equivalent in these works to the use of compound instructions in the current work, which is equivalent to several tasks on each part of the image, but it is not able to do all of the tasks at the same time and it is not able to process a compound instruction, unlike in this study.

1.3 Goals of the current work

In this work, we will learn and perform multiple visual tasks. The learning will be mostly one task at a time, but during image analysis, we will perform multiple tasks together. The three main reasons why we chose this

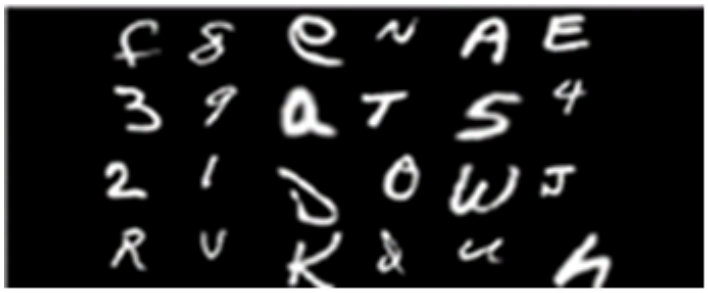
specific set-up of work in which we learn tasks individually:

1. Complexity of the learning
2. Combinatorial generalization
3. Learning Different tasks at different times.

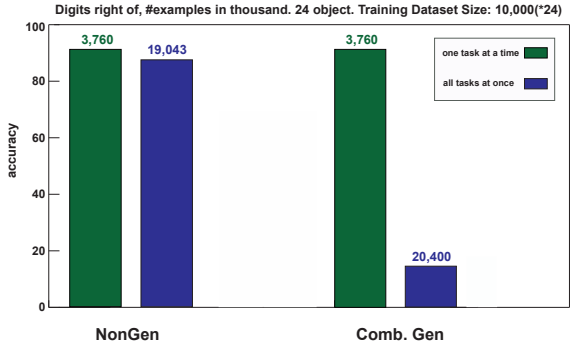
The first issue concerns the problem of learning to perform combinations of different tasks. If we want to learn all possible combinations of tasks, [training] would take a long period of time and would require many resources in order to achieve this goal. In the experiments performed in section 3.5, we saw that without training on every single example, a neural network could not generalize and perform several tasks at the same time while trained on only one task at a time. Getting all the possible combination of examples is not always a feasible task, and even if we do go through such a large number of examples, it would be costly in terms of time and number of examples. The second reason comes from limitations on combinatorial generalization when multiple tasks are learned simultaneously. Combinatorial generalization is the type of generalization where we expect the model to be able to learn all possible example combinations when it was exposed to only some of them. For example, if we have a dataset with images where, in each image, we have a list of characters. As demonstrated in fig 1a, the number 3 can be on the the left of the number 9 in an image, or 9 can be on the left of 3. Those are two exaples of possible combinations of the numbers 9 and 3; however, typically, the model cannot know how to perform unseen combination until it has seen all possible combination examples. This means we assume that, upon seeing only one of those combinations, a model using combinatorial generalization would be able to learn to do all other combinations without needing to learn all other possibilities. As such, given only the example of the number 3 on the left of 9, it would be able to understand 9 on the left of 3, meaning it was able to generalize on the combinatorial level.

Previous work on combinatorial generalization limitation has shown that hat if multiple tasks are learned together, for example, to extract all the properties of two or more different objects in an image, it is possible to learn to carry out the task, but the learning does not generalize to new object-property combinations. An example (taken from (15)) is shown in figure 1b below

The example shows that learning multiple tasks together can lead to a substantial reduction in combinatorial generalization. This example compared two variants of the Counter Stream model, which is described in section 2.1. In the first variant we learn all the tasks together and test for generalization using a technique called Read Out. Using this technique, we read out results on novel combinations from a model trained on a limited set of combinations, as described in more detail in section 3.3 below. In the second variant, we learn one task at a time in the usual manner. As a dataset, a subset of EMNIST[15] dataset was used with a set of 29 characters. Given an image with objects (letters and digits), the task is to find the object to the right/left of a selected digit. For each of the model variants, two experiments were compared. In the first, the model was trained and tested in a standard manner, without testing combinatorial generalization. In the second, the model was tested specifically on combinatorial generalization, by excluding some of the digit-relation combinations during training (e.g., 7 to the left-of 3), but testing them during test time. As can be seen from the results, the model which is trained on all the examples at once, does not perform well on combinatorial generalization problems.



(a) The EMNIST variant which was used for the experiment.



(b) Results when learning one task at a time vs all tasks at once, 'NonGen' is when performing without combinatorial generalization and 'Comb Gen' is combinatorial generalization.

Figure 1: Combinatorial Generalization past work.

The third reason for not learning all the tasks simultaneously is that training different tasks may become available at different times. A single training image will contain only a limited subset of the tasks that the system needs to learn, and new training images may contain a limited set and even a single novel task. On the other hand, during the analysis of a novel scene, we would like to perform multiple tasks to minimize the overall image analysis time. We show that in our model it is possible to instruct the model to learn selected individual tasks, and during the image analysis, perform selected subsets of tasks simultaneously without any additional training. In addition, learning one task at a time reduces significantly the combinatorial generalization problem, but this will not be the focus of this work. The discussion above motivates the current work, , but the work focuses not on combinatorial generalization, but on the general issue of learning the tasks sequentially and subsequently executing more than a single task at a time.

2 Methods

This section will describe the databases which were used, as well as the model architecture we used and its variants.

2.1 The Counter Stream Model

The Counter Stream Model (CS) is the model architecture we used in all of our experiments. This model allows us to learn selected tasks, one at a time, and later execute either single or multiple tasks together. It is composed of three parts: Bottom-Up1(BU1), Top-Down(TD) and Bottom-Up2(BU2), and can be used with almost any standard neural network as a backbone (See figure 2).

The BU1 part is a standard neural network (The backbone itself). The main difference is that after a Resblock,

we save the information for the TD part. The BU1 pillar receives an image and outputs a multi-label classification of all of the objects in the image that it received.

The next part is the TD, which is something like the reverse of the BU pillar, instead of inputting an image, we give instructions initially for both a task to perform and an argument, what to apply the task to. This is given initially in a symbolic, one-hot representation. During learning the system learns an 'embedded' vector representation for the selected task.
The TD will increase the resolution until we get an image which is usually used for segmentation tasks on the object we asked about. The main goal of this pillar is to assist the next BU2 stage on the object and the property of this object we are asking about, and The property might be something like colour, shape of the object or something about its surroundings, such as its neighbours.

Lastly, we have the BU2 as the last part. This part is similar to the BU1 part, but instead of using just the image, it will use the information it gains from the TD part about the target and the instruction we need. It will use this information and output a new vector for classification/multi-class classification of specific properties we asked about, that are related to the object.

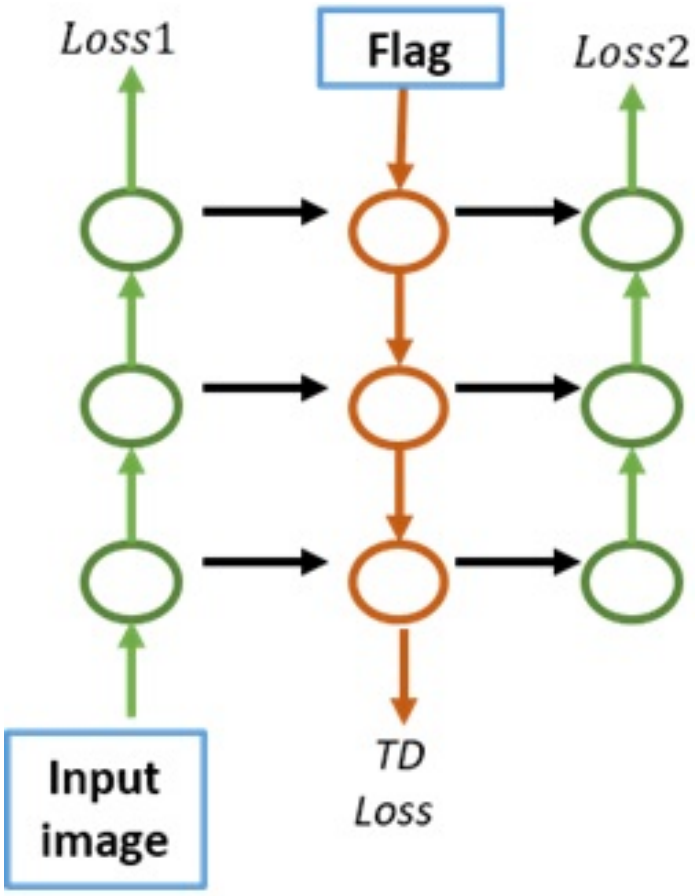


Figure 2: A simple sketch of the counter stream network, with the flow of information.

2.2 Datasets

2.2.1 Squared MNIST and Row MNIST Dataset

The squared MNIST is made up of 4 digits arranged in a 2x2 grid of numbers between 0 to 9 from the MNIST dataset LeCun et al. [16]. Each character has a resolution of 28x28, and the full image has a size of 56x56 (See figure 3). Another version which we used is called row MNIST which has a number of digits in a row, usually we use the 8 digit version, which has size of 28x112(see figure 4).

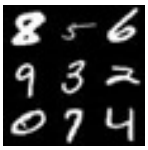


Figure 3: Squared MNIST dataset.

Figure 4: An example of row MNIST with 6 digits.

2.2.2 Avatars Dataset

The Avatars dataset is composed of six different avatars(example in figure 5). In each image, four of those avatars appear with different properties in each image, such as their skin colour, glasses, shirt, hair colour etc. In total, each avatar has seven different properties, and each property has as many as eight different options. Each image also has a different background which may be composed of waterfalls, beaches and so on. (See figure 5) This dataset can be used for classification, segmentation and object relation, and it was used as the primary dataset for most of the experiments which we conducted.

3 Experiments and Results

In this section, we describe the experiments which were conducted in order to study the capacity of the model when learning to perform one task at a time, and then combining several tasks to perform them simultaneously. In these experiments, we used the Avatars dataset and the row and rectangle MNIST datasets. The learning process in these experiments is done one task at a time by selecting a task and an argument. We then combined multiple instructions together, by using their one-hot superposition, when testing the results of the combined instructions.

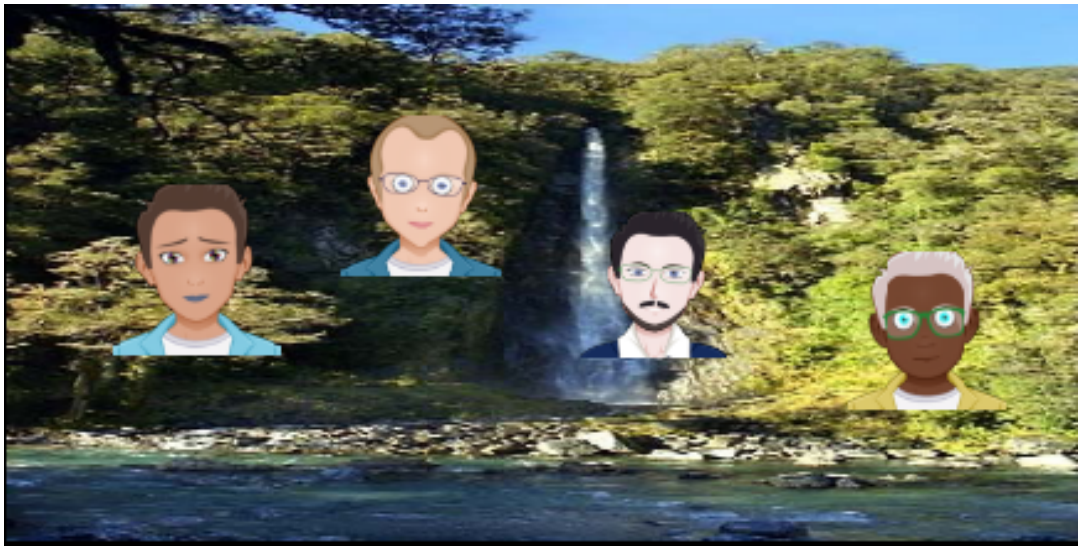


Figure 5: An image from the avatars dataset.

3.1 Object properties: same object, multiple properties

The following experiment was conducted on the Avatar database, and its goal was to extract one property at a time, and then extract multiple properties at once and maintain a high accuracy on a new task which is composed of few tasks together. While training we asked for a specific avatar in the image and a property of this avatar which we were interested in. The representation of the instruction was composed of 2 one-hot-vectors, we can call this a "compound instruction", since it is compound of two one-hot-vectors. The first one-hot vector reflects the id of the avatar which we want to ask about, for example, for the second avatar we would get the instruction 010000. The second one-hot vector is the property that it was asked to identify and is created similarly. We concatenated those two one-hot-vectors as can be seen in Figure 6a, and then sent it as an input to the Top-Down part of the Counter Stream network. As an output, we expect to get the property we asked for.
For this experiment we had 800 images for the training phase and 200 for the test phase, for each image we randomly asked a question regarding one avatar and one feature, which meant that potentially we had 22400 different examples during training and 5600 at test time. The training was composed of 700 epochs with the sampled examples, we used 3 losses for this experiment for each of the pillars of the Counter Stream model. For BU1 we used binary cross entropy loss for finding the avatars id. For TD we used mean square error loss for the segmentation of the avatar we asked about, and for BU2 we use cross entropy loss for the property we asked about

After we finished training, we began the testing phase in which we used the same procedure in order to gain properties(one-hot vectors). We began by asking for a specific property from one avatar(as we did during training), in doing so we achieved 96% accuracy, this achievement is impressive in itself, since we proved that we can train one model for different tasks using 1-hot-vector to specify the task we want. Going further we increased the number of tasks, instead of asking for one property, we asked for two properties, using the same model that we trained before. For this we changed the property vector, by adding 1 in two spots instead of one spot, so we get a vector of the type: 100001, where the first spot represents the first property of the avatar and

the last spot represents the last property. This means that since we have 21 options for the combination of digits, we had 16,800 different options for examples. When testing this case we achieved 94% accuracy.
Based on these results we decided to go further and try to identify all the properties at once using this model. This time the property 1-hot-vector was filled with 1's (e.g. 111111). This way we could inquire about all the properties which are possible at once, as can be seen in Figure 6b.
Since we have only one option for each avatar(all features), we have 800 examples. When we were testing this experiment we still achieved 94% accuracy which is the same drop we had with two properties.
Besides our testing with extracting all the properties of a selected avatar, we also tested the ability of the model to perform combinatorial generalization on compound instructions. During training, some of the person-property pairs were excluded from the training set. For instance, the combination avatar-3 with glasses-type 5, was never shown in training, but was tested at test time, as a part of a compound instruction. The performance of the model on the new combinations was not different from its performance on trained person-property pairs. Combinatorial generalization with compound instructions is further discussed in section 3.4 below.

Summary of results:

Testing a single property at a time: accuracy 96%
Testing 2 properties together: accuracy 94%
Testing all properties together: accuracy 94%
Testing combinatorial generalization: accuracy 94%

(a) The representation of the vector in the training phase. (b) The representation of the vector in the testing phase.

Figure 6: The use of 1-hot vector in the experiments.

3.2 Object properties: same property, multiple objects

In the last experiment of **Object properties: same object, multiple properties** we tackled the problem of having to deal with multiple tasks at the same time. Moving forward from there we wanted to look at the problem of dealing with multiple objects at the same time. This time instead of dealing with the first part of the compound instruction, which is responsible for the tasks, we changed the second part of the compound instruction, where we decided on the objects that we wanted to look at.
The experiment was in the same manner as the **Object properties: same object, multiple properties** experiment, but this time we changed the one-hot-vector of the first vector which represents the avatar id's, so instead of having 1 on only one spot, we had it in more.
For this experiment we used the same initialization of the Avatars dataset of 800 images for the training phase and 200 for the test phase, in each image we asked randomly for a question regarding one avatar and one feature, which means that potentially we had 22400 different examples during training and 5600 at test phase. The training was composed of 700 epochs with the sampled examples, we used 3 losses for this experiment for each

of the pillars of the Counter Stream model. For BU1 we used binary cross entropy loss for finding the avatars id. For TD we used mean square error loss for the segmentation of the avatar we asked about. For BU2 we used cross entropy loss for the property we asked about.

Looking at one avatar and one feature at a time we achieved the same results that we mentioned in the experiment before (96% accuracy). Then we tried to look at all the avatars at once, for this we had 1400 different examples to test and the performance of our model on all the avatars at once was lower then before, achieving only 75%.
We considered the possibility that the lower accuracy was caused by incorrect paring of objects and properties. To test this possibility, we applied a test we call ”confused accuracy”, in which we accept an output of a feature if it is correct for one of the four avatars we selected. For example, if avatar number one has a blue shirt, but the model said that avatar four had the blue shirt, we still accepted this result. For this we used the same 1400 different examples which we used to test all the avatars simultaneously,in doing so the accuracy increased to 86%.
We can see that we still have a significant decrease compared to the 94% we had before, but an increased accuracy results compared with the 75%. From these results we propose that the reduced accuracy could arise from a combination of two difficulties. The first one might be with focusing on all parts of the image at the same time, since the relevant avatars for a given property were located at different regions of the image. The second problem that the model is having is the binding problem, where the model is having an issue to bind between one property and the avatars in the scene.The reduced accuracy due to binding can still be of value especially for tasks that do not rely on correct binding, for example, finding which glasses are the most popular in a given scene.

Summary of results:
Testing properties of a single avatar: accuracy 96%
Testing one property for all the avatars: accuracy 75%
Allowing for incorrect binding: accuracy 86%

3.3 Combining spatial relations

The purpose of this experiment was to find which digit is on the left, right, top, or bottom of another digit. We trained the Counter Stream network by giving it a digit and a selected direction, after which the network had to classify which number is in the given direction. For example, if we give the instruction, ”Above,1”, and the number 3 is above 1, we expect the network to output the number 3. In the event that the digit is in a corner and there is no digit on the left/right/top/bottom, we will expect the number 11 which means that there is no digit in this direction.
For this experiment we had 400 images for the training phase and 100 for the test phase. For each image we randomly asked a question regarding one digit and one direction, in the event that the digit was in a corner, we

expect the number 11 as an answer, to represent that there is no number in that direction.
We split this experiment into two parts, the first using a common representation of the output digits and the second separate representations. In the common representation we have a single output vector , which represents the results for the left and right neighbour, in a vector with a length of ten.
To test the model with common representation on tasks that involve compound instructions, we used a method called Read Out, which allows us to test more than a single output. To use this method we train the model to answer one question at a time using a single task instruction. Each time, the model is trained for one task, until it is fully trained. Now, when we want to carry out two tasks at the same time, we first freeze the model and add another fully connected layer to the last vector. Those layers are trained to identify the specific left or right neighbour, while the TD model instruction is for both the left and right neighbours at the same time. If our model can identify which digit is on the left and which is on the right with this one representation, the model is able to learn everything with a common vector which represents both sides.

This experiment was done using the same type of row MNIST as in Figure **??**, this task involved 400 training images and 66 testing images. In total, we have 2400 examples for training and 396 for the test phase. We did 100 epochs for this experiment, and more 1400 epochs for training the Read Out layers. So, in total, the Read Out was trained for 1500 epochs, while the regular separate representation had 100 epochs. We used three losses for this experiment when learning the weights of the model and an extra loss for each of the additional fully connected layers of the Read out (in total). For BU1 we used binary cross entropy loss to find the numbers in the image. For TD we used mean square error loss for the segmentation of the digit about which we asked about. For BU2, as well as the two fully connected Read Out layers, we used cross entropy loss for the digit in the direction about which we asked.

Doing so we achieved a low accuracy rate, with 67% accuracy for the first direction Read out and 56% accuracy for the other direction Read Out. We suspected this issue arises from the binding problem discussed above. This time we checked for confused accuracy, to see if they identified the correct property but had the wrong direction; in that case, we still accepted the result as valid. The results confirmed our hypothesis, and the accuracy in both sides increased to 94% accuracy. This confirmed the binding problem discussed in section 3.2.

For the second part of the experiment we compared the results of common representation to those of separate representation, where each side has ten neurons (20 neurons total), in comparison to the common representation which had ten output neurons only. As described in section 3.1, after training on one direction, we tested on combined instructions which ask for both left and right at the same time. In total we have 12,800 different examples in the training phase and 3,200 in the test phase which could possibly be sampled during 75 epochs of the training process itself. We used three losses for this experiment for each of the pillars of the Counter Stream model. For BU1 we used binary cross entropy loss for finding the numbers in the image. For TD we used mean square error loss for the segmentation of the digit about which we asked about, and for BU2 we used cross entropy loss for the digit in the direction about which we asked about.

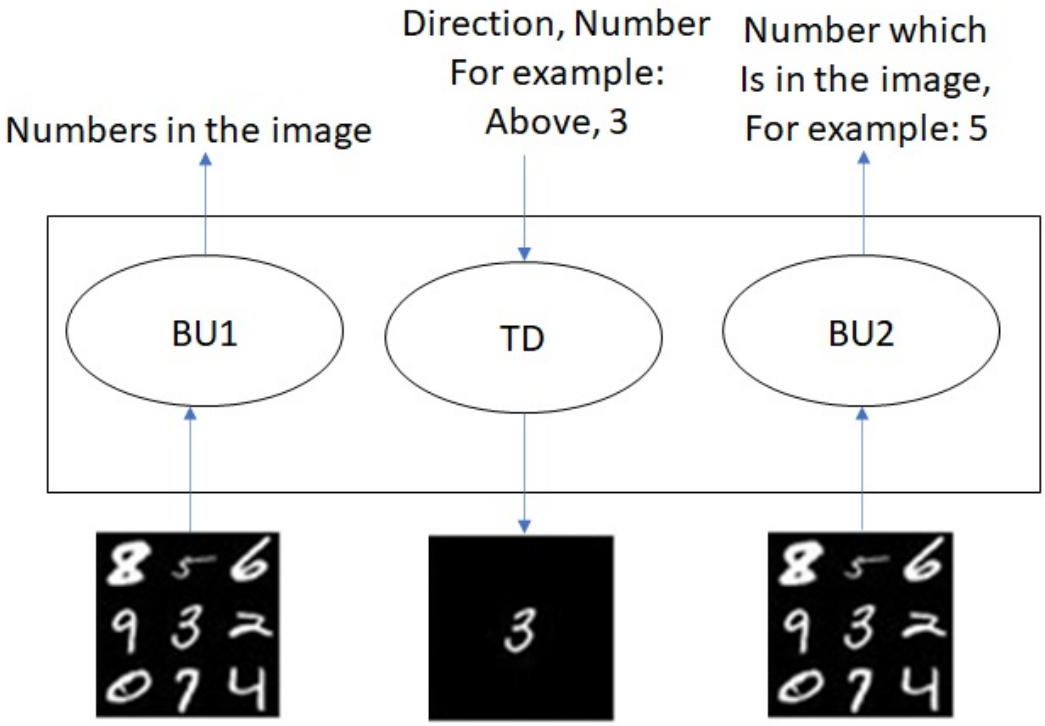


Figure 7: the training of the model, the testing process for the first experiment.

With those settings we achieved about 94% accuracy. When training the model we can again see success in task-related training as we have seen before, where we use the separate representation with combined instructions to solve the binding issue seen with common representation.

Summary of results:

	one property	two properties	two properties with confusion
common representation	96%	61.5%(average of 56, 67)	94%
separate representation	96%	94%	-

Following those results, we conducted another experiment in order to check how common representation performs on problems that require combinatorial generalization versus the performance of separate representations, as we will see in the next section.

3.4 Combinatorial generalization

The goal we want to achieve in this experiment is to check the combinatorial generalization ability which we discussed in section 1.3, and of our results in section 3.1. During this experiment we tested combinatorial generalization for both common and separate representations. For this experiment we use the row MNIST and we have two types of instructions: $right\_of$ and $left\_of$. The task of the model is to find which number is the neighbour of a given reference number in the specific direction. In the case of no neighbouring number (border digit) we expect the number 11 as an answer (which represents that there is no neighbouring digit). During training, we use each pair of digits with only one type of instruction and not the other. In this combinatorial

generalization test, a subset of combinations (a digit and its neighbour) are excluded during training. In particular, for some of the digits we excluded in training all the neighbours on one of the two sides. For instance, the digit 7 will be trained on extracting its left neighbours, but not right neighbours. During testing, we test combinatorial generalization by testing specifically combinations not used in training. This experiment was performed with separate and common representation in order to compare between the two. We used the exact settings that we had in section 3.3, using the same type of row MNIST as in Figure 4, on both training and testing we ask about one side at a time. This task involved 400 training images and 66 test images, in total we have 2400 examples for training and 396 for the test phase. In total we performed 100 epochs for this experiment. We used three losses for this experiment when learning the weights of the model. For each of the pillars of the Counter Stream model. For BU1 we used binary cross entropy loss for finding the the numbers in the image. For TD we used mean square error loss for the segmentation of the digit we ask about and for BU2, we use cross entropy loss for the digit in the direction we asked about.

Doing so the results on combinations which were not seen before, were the following, the separate representation accuracy was reduced to around 52%, while the common representation accuracy was 86%, those results show that with a common representation we get better combinatorial generalization than the separate representations, since the separate representation has different neurons dedicated just for a specific side, so in case that this side was not introduce to specific number it does not perform well, while the common representation network has the same neurons for both left and right sides, so if a specific digit was introduced to one side only, the neural network would know how to handle it when it is shown to the other side.

Summary of results:
separate representation with combinatorial generalization: accuracy 52%
common representation with combinatorial generalization: accuracy 86%

3.5 Changing lateral connections

In order to better understand and examine how the Counter Stream network achieves the results we have seen with compound instructions, we examined the role of the lateral connections in Counter Stream network. We tried to repeat the 'Object properties: same object, multiple properties' experiment but each time we eliminated one of the two sets of lateral connections. The goal was to see how the lateral connections affect the accuracy when trying to identify all properties at once, after training on one property each time.
For all the experiments we used the same initialization as the previous experiment using the avatar dataset, where we have 800 images for the training phase and 200 images for the test phase. This gave 22400 different examples during training and 5600 at test time, with 3 losses. After BU1 we used binary cross entropy loss for finding the avatars id. For TD we used mean square error loss for the segmentation of the avatar we selected, and for BU2 we used cross entropy loss for the property we selected.

We used two baselines for the Counter Stream, one is a baseline with two lateral connections and was based

on Resnet-18 as a backbone and the other was based on VGG-19 which does not have lateral connections as a backbone. Each of these was trained for about 700 epochs.
The results showed a clear advantage of Resnet-18 backbone, which achieved 94% accuracy on the test phase compared to the VGG-19 backbone, which achieved only 71% when trying to identify all features at once.
We then continued in order to check what happened if we did not use lateral connections in Resnet-18 based Counter Stream model. This time we achieved 84% and as we expected, the lateral connections had an important role in the ability to generalize from a single task during learning to multiple tasks using a compound instruction.
We then tested more specifically the two sets of lateral connections. We first used only the BU1-TD lateral connections (without TD-BU2). We trained the model with about 750 epochs to maximize the accuracy and we achieved 85% accuracy of identifying all features at once.
Lastly, we tried to use only TD-BU2 lateral connection (Without BU1-TD) and this time we achieved 92% accuracy when trying to identify all the features at once even though we trained in only one feature at a time. From these results, we can see that the TD-BU2 set of lateral connections play the main role in generalizing from a single task to performing multiple tasks simultaneously.

Experiment / Model	Baseline (1 feature)	all connections	No-Connections	BU1_TD only	TD_BU2 only
VGG-19	97%	-	71%	-	-
Resnet-18	96%	94%	84%	85%	92%

Table 1: Results of changing the lateral connections.

3.6 Comparison to Channel Modulation

In addition to our BU-TD model, there is in the literature another example of using task-selection, termed Channel Modulation[12]. We compared next the ability of this model to generalize from a single to multiple tasks. Task selection is obtained in channel modulation by multiplying each channel in the model by a task-dependent weight. The set of modulation weights are for each task a vector learned during training. Each task is represented by a one-hot vector, which then generates the vector of leaned weights, as illustrated in Figure 9. For this experiment, we trained the model on one avatar and one property at a time, with the same setting as we used in the Object properties: same object, multiple properties experiment in Section 3.1, with a baseline of Resnet back- bone Counter Stream. The Avatar database was used with 800 images for the training phase and 200 for the test phase. The results of this experiment (as described in section 3.1) were 96% accuracy when testing on one property, When we tested on all the properties simultaneously we obtained 94% accuracy.

The next phase was to perform the same experiment on the channel modulation. We used exactly the same training as for the BU-TD experiment. The model was constructed from Resnet as a backbone, with channel modulation for each channel. We had cross entropy loss at the end of the network for the selected property and avatar of the given task. We used one-hot-vector in the same manner as we did in all of our other experiments. To create compound instructions, we combined the representation of one or more one-hot vectors of the individual

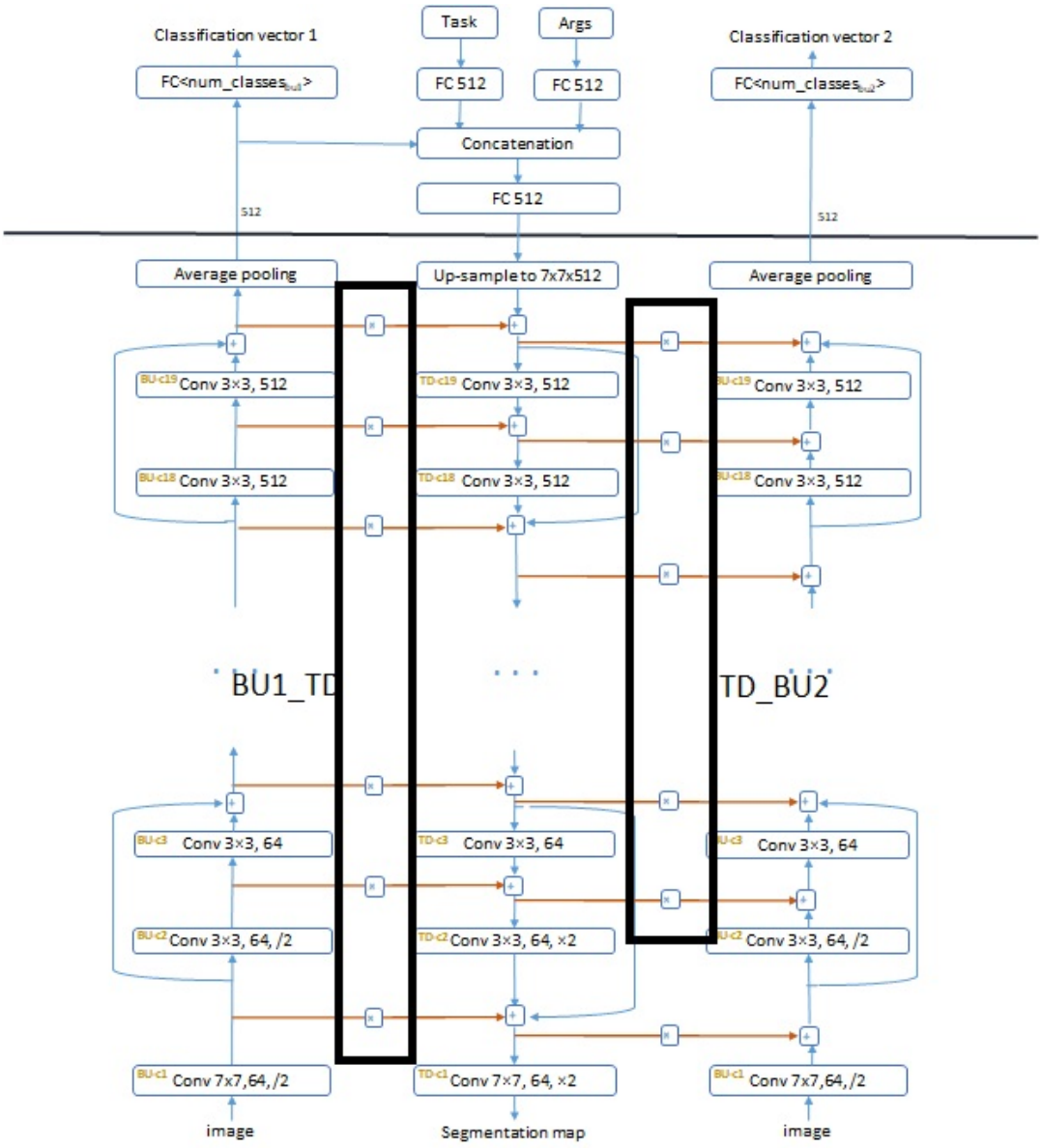


Figure 8: The architecture we used with Resnet-18 as a backbone, with the BU1_TD and TD_BU2 lateral connections marked.

tasks

In testing the channel modulation model, we achieved 88% accuracy on one feature at a time. When we tested the model on performing the task of extracting all the properties at once, accuracy was 56% in comparison the BU-TD 94% accuracy. As discussed in the introductory part, the ability to learn one task at a time, but perform multiple tasks together when possible is a highly desirable property when dealing with a large number of tasks. We conclude that the ability to perform such task combination depends on the way in which tasks are selected and combined, and that the counter-streams model shows a better ability at task-combination compared with channel modulation.

	Channel Modulation	Counter Stream
One task at a time	88%	96%
combining tasks	56%	94%

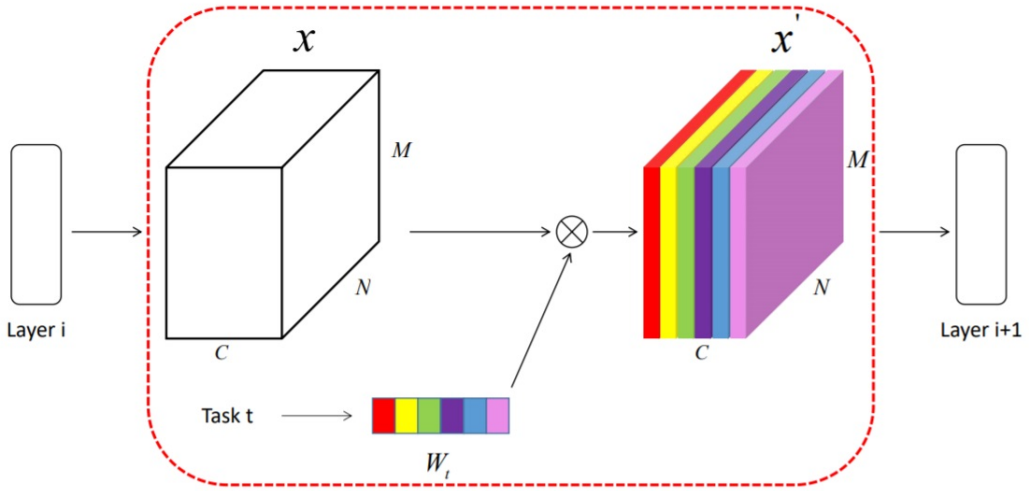


Figure 9: The Channel modulation single module architecture as can be found in ref[12] .

4 Summary and Conclusions

In this work, we have demonstrated a model which learns to conduct one task at a time during the training process, and subsequently it is able conduct two or more tasks simultaneously using the Counter-Stream architecture with a combination of TD instructions. The experiments showed that it is possible to generalize without any additional training to multiple tasks but using compound instructions. In the avatars dataset, the model can extract not only two, but all the properties of a selected avatar by combining together all the instructions for the individual properties. We also showed that our model is the state of the art, and get better results than the channel modulation in the generalization task of learning one task and performing all the tasks at once.

The main limitation we found in combining tasks is not the number of tasks, but the so-called binding problem. The problem arises when the model combines two different tasks, f and g to compute $f(x_1) = y_1$ and $g(x_2) = y_2$: the model may correctly extract (y_1, y_2), but fail to perform the correct association between the task and the corresponding output value. We showed that the binding problem can be avoided, and consequently using compound tasks becomes possible, but using an appropriate separation of the target representations (as in the case of left-neighbors and right-neighbors). However, we have shown that , the separation of the target domains has a price, as combinatorial generalization becomes more limited. We conclude that it is possible to combine tasks more broadly by using separate representations, or achieve high combinatorial generalization using a common representation, but not botn.

The results show that the analysis of a scene, which requires the combination of a large number of tasks, can proceed by a sequence of steps, each composed of a number of tasks performed simultaneously. The tasks can be acquired individually, and then combined with no additional learning, by using compound task-instructions. The main limitation of compound tasks we encountered were imposed by the binding of the different outputs to

the corresponding tasks. The results also show that depending on the problem at hand, it is possible to tradeoff between the ability to perform multiple tasks together, and the ability to generalize to novel combinations.

In regards to the implications of the results, we can see several use cases for this mode of processing as a building block for complex systems, which require learning a large number of tasks, and performing multiple tasks together, while maintaining high combinatorial generalization ability. It could be used in the medical field, when the ability to learn different tasks at different times and the ability to combine them while maintaining combinatorial generalization are necessary. Other complex vision system in current use such as autonomous vehicles and smart drones also deal with combinations of different scenes and many different tasks to be learned, and have to deal novel combinations. Such complex system could benefit from combining tasks within the limits of the model which we discussed above.

Regarding future work, we should consider going further, working with more natural image datasets, and solving some of the issues which force us to choose between separate and common representation, mentioned in the beginning of this section. Lastly, we would like to explore new ways in which we can deal successfully with the binding problem explored in the Object properties: same property, multiple objects experiment.

5 Literature

[1] X. Danfei, Z. Yuke, C. Christopher B., and F.-F. Li, "Scene graph generation by iterative message passing," *IEEE CVPR*, pp. 5410–5419, 2017.

[2] G. Ross, D. Jeff, D. Trevor, and M. Jitendra, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE CVPR*, pp. 580–587, 2014.

[3] G. Ross, "Fast r-cnn," *ICCV*, pp. 1440–1448, 2015.

[4] R. Shaoqing, H. Kaiming, G. Ross, and S. Jian, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv : 1804.05810*, 2016.

[5] R. Joseph, D. Santosh, G. Ross, and F. Ali, "You only look once: Unified, real-time object detection," *CVPR*, 2016.

[6] R. Joseph and F. Ali, "Yolov3: An incremental improvement," *arXiv : 1804.02767*, 2018.

[7] S. Ozan and K. Vladlen, "Multi-task learning as multi-objective optimization," *NIPS*, pp. 525–536, 2018.

[8] A. Netanyahu, "Cyclical bottom-up top-down neural networks for relational reasoning," Master's thesis, Weizmann Institute of Science, 2018.

[9] A. Yaari, "The counter stream model for image object to attribute relation classification," Master's thesis, Weizmann Institute of Science, 2018.

[10] L. Hila and U. Shimon, "Multi-task learning by a top-down control network," *arXiv:1410.5093*, 2020.

[11] U. Shimon, A. Liav, and V. Ben Zion, "Image interpretation by iterative bottom-up top-down processing," 2020.

[12] Z. Xiangyun, L. Haoxiang, S. Xiaohui, L. Xiaodan, and W. Ying, "A modulation module for multi-task learning with applications in image retrieval," *arXiv:1807.06708*, 2018.

[13] J. Saumya, L. Nicholas A., L. Namhoon, and T. Philip H.S., "Learn to pay attention," *arXiv : 1804.02391*, 2018.

[14] X. Kelvin, B. Jimmy, K. Ryan, C. Kyunghyun, C. Aaron, S. Ruslan, Z. Richard, and B. Yoshua, "Show, attend and tell: Neural image caption generation with visual attention," *ICML*, pp. 2048—-2057, 2016.

[15] C. Gregory, A. Saeed, T. Jonathan, and S. André van, "Emnist: an extension of mnist to handwritten letters," *arXiv : 1702.05373*, 2017.

[16] Y. LeCun, C. Cortez, and C. C. Burges, "The mnist handwritten digit database."

6 Acknowledgements

Firstly, I would like to sincerely thank my M.Sc. Advisor, Professor Shimon Ullman, for his guidance and support in the last year. The discussions we had during that time were truly inspiring. The ideas he brought up really contributed heavily to the foundation of this thesis, and without him it would have not been possible. It was a great honor to have worked with him. I would also like to thank the faculty of Mathematics and Computer Science at the Weizmann Institute of Science for their support and help during the last two years. Lastly, I would like to thank Liav Assif, for the advices, help, and results of past works which he shared with me and helped, and to Hila Levy, for her help and advice during many parts of the thesis.