
Data Analytics
Stiftung Universität Hildesheim
Marienburger Platz 22
31141 Hildesheim
Prof. Dr. Dr. Lars Schmidt-Thieme

# Seminar Data Analytics III
# One Model To Learn Them All
# Semester 3

Abdul Rehman Liaqat

271336, Liaqat@uni-hidesheim.de

**Abstract**

With the help of deep learning, it is possible to outperform humans in certain task i.e. image recognition. This incredible feat comes with a price of spending a lot of time selecting and fine tuning an architecture for specific problem hence for each task a specialized deep learning architecture is needed. Authors purpose a solution to this problem by designing a single architecture which is trained for each task with their respective type of input and output. Also this multi-model architecture, as authors call it, is composed of important modules,such as convolutional layers, attention-mechanism layers and sparsely-gated layers, which are necessary for good performance in each task . As a result, multi-model performs quite well for problems with less data, when trained with the data from other tasks. Also the performance for other tasks only degrades slightly.

# Contents

# 1 Introduction

Human brain have a common area which is triggered whenever a banana is seen, heard, written or spoken. Of course there are very separate specialized areas too which are lit up differently according to the action described earlier. This commonness leads to a question: Just like a human brain

> Can we create a unified deep learning model to solve tasks across multiple domains?

**One Model to Learn Them All** presents this question and try to answer it with a deep learning based modular architecture. This multi-model architecture ,as authors called it, is tested against intrinsically different type of problems such as image captioning, language translation and image recognition. The model perform well in general and outperforms competitors for problems with less dataset. An example from the paper is:

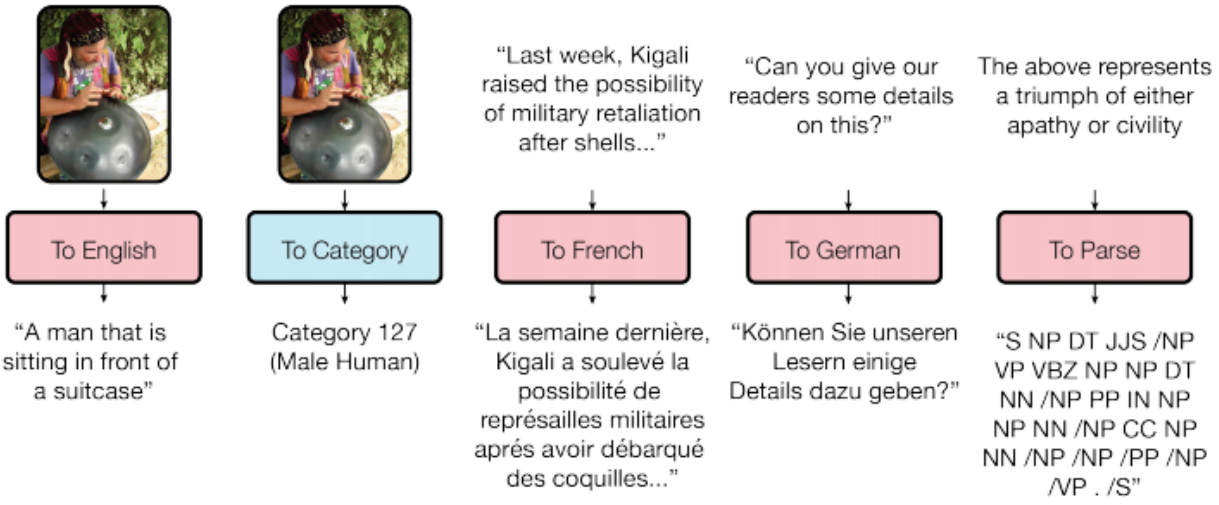


Figure 1: Examples decoded from a single MultiModel trained jointly on 8 tasks. Red depicts a language modality while blue depicts a categorical modality. [4]

Continuing from the original question posed in the beginning of this section, further questions can be raised about:

1. What is the benefit of having one model for all tasks?
2. Intrinsically, which qualities are in place when we train one model for multiple tasks?
3. What is the most intuitive way of going about using one model for all tasks?

Starting from the first question, one obvious benefit is the saving of time and resources. When only one model is used, training for one task helps in training for another task through intrinsic transfer learning. Common parts within different tasks takes help from the training of same part. Lastly, the most intuitive way would be to convert all kind of inputs to a same format before further treating through all stages. That is exactly the method proposed by the authors. All type of input or output is first convert from or to required format respectively, through Modality nets.

# 2 Multi-Model Architecture

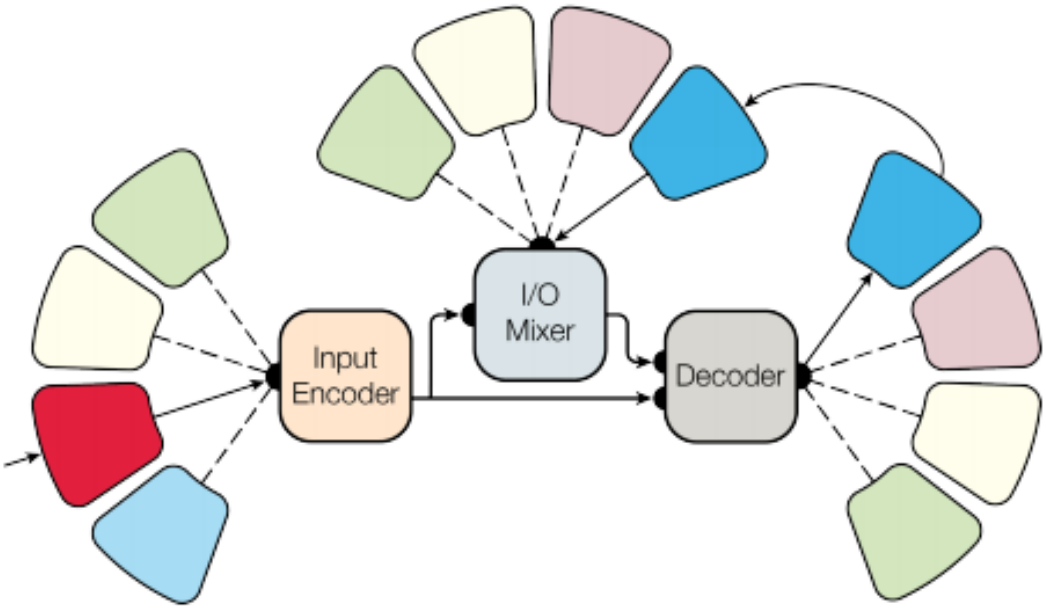


Figure 2: The MultiModel, with modality-nets, an encoder, and an autoregressive decoder [4]

Multimodel architecture is compose of different modules.

## 2.1 Components

There are four basic modules of the whole multimodel architecture:

1. Modality Nets
2. Encoder

3. I/O Mixer

4. Decoder

Each of these modules are composed of Convolutional blocks, Attention blocks and Mixture-of-Experts blocks.

## 2.2 Component Models Architecture

Now looking into the architecture of main modules.

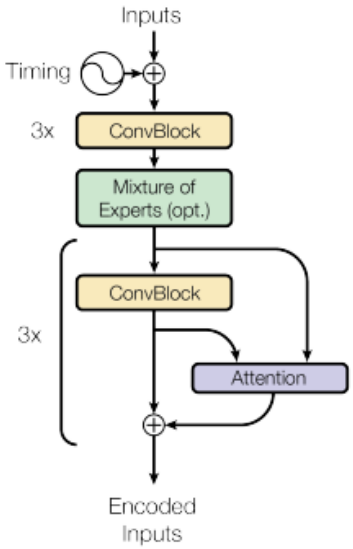


Figure 3: Encoder Architecture. [4]

Input encoder takes the output of Modalities as input and process them. Each input to Encoder is time stamped with a vector of time. Afterwards it pass through three layers of ConvBlock (figure 8). ConvBlocks output is fed into Mixture of experts and then comes three blocks of ConvBlock and Attention modules with skip connection. Lastly, output of last ConvBlock is summed with last Attention module and thus input coming from modalities is encoded.
Output of input encoder is fed to I/O mixer and decoder.

Output of encoder and modalities with auto-regressive connection of decoder as input is fed to I/O mixer. This helps in providing auto-regressive capability to our architecture which means that multi-model architecture also learns about the patterns hidden in temporal

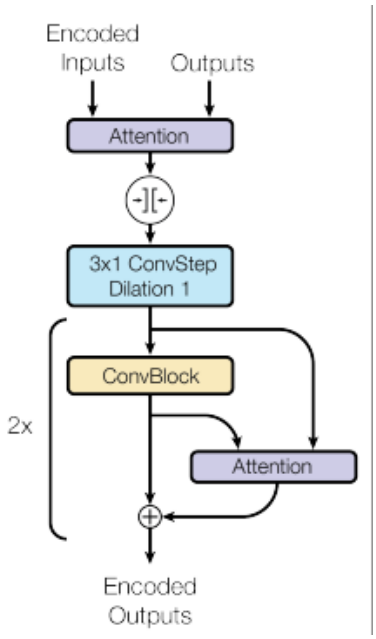


Figure 4: I/O mixer architecture. [4]

sequence. In other words, multi-model architecture knows how much weight should be given to the past values. How much near past is important to distant one.
I/O mixer takes all input to attention module which is then padded by zeros on the left. This helps in prohibiting I/O mixer in accessing future values. Next is a ConvStep following by two block of ConvBlock plus Attention module. Output of last ConvBlock and Attention module is summed. The final ouput is fed to decoder.

Decoder takes in the output coming from input encoder and I/O mixer. Both are concatenated and fed to two ConvStep blocks followed by four ConvBlock plus Attention module combination. Other input of of Attention module is Encoded inputs as shown in figure 5. The output of final ConvBlock and Attention module is summed and fed to modalities which will transform the decoded output to required output format.

## 2.3 Base Component Architecture

As noticed above, each of the module of multi-model architecture is composed of small components. Each component has it's importance to solve one specific kind of task for example attention block is good for tasks related to natural language processing. Similarly ConvBlock is good for Image related tasks. Also since all of these components are used in each task it is tested that if unrelated components to one task has any positive or negative impact on the performance. Results are shared in the experiments sections. Base components are following:

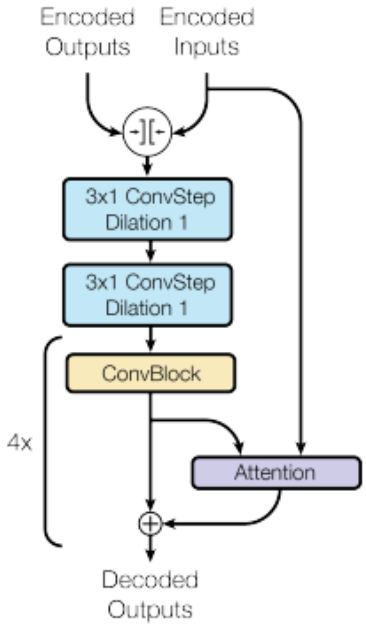


Figure 5: Decoder architecture [4]

1. Attention Block
   (a) Dot-Product Attention
   (b) ConvStep
      i. SepConv
   (c) Pointwise Convolution
2. ConvBlock
3. Mixture of Experts (MoE)

Here one noticeable thing is that ConvStep is also used in other components such as ConvBlock.

Attention Block: As described earlier, this block has proved to be efficient for natural language processing tasks. Attention block helps identify the part of language which are to be considered more hence named attention. In each attention block, the input is time stamped and then fed to two ConvStep components with $5 \times 1$ filter size and having dilation value of 1 and 4 respectively. Output of last ConvStep is fed to Dot-Product attention block. In parallel, input to the attention block is fed to two Pointwise convolutions which are followed by a Dot-Product attention block. Final Dot-Product attention block takes input from the two Pointwise convolution blocks and Dot-Product attention block.

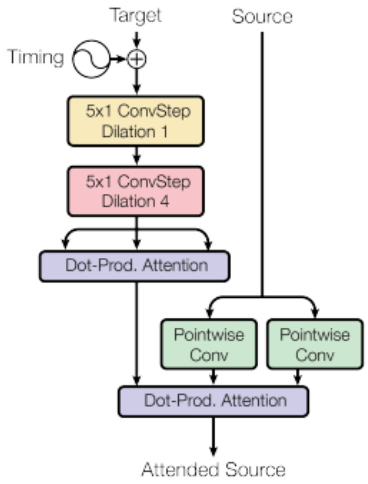


Figure 6: Attention Architecture. [4]

Dot-Product Attention Architecture: Dot-Product attention architecture, consist of three inputs called Query(q), Attention Keys(k) and Value keys(v) respectively. Each input is represented as a matrix of numbers. First query and attention keys product is multiplied and passed through a softmax operation. Thus only useful information passes to the next step. Finally value keys and the softmax output is multiplied which results in Dot-Product attention output.

ConvStep: A ConvStep can be defined as :

$$ConvStep_{d,s,f}(W, x) = LN(SepConv_{d,s,f}(W, ReLU(x))) \tag{1}$$

where:

| | |
|---|---|
| $d$ | = dilation factor |
| $s$ | = stride |
| $f$ | = input feature depth |
| $W$ | = weights of size $hxw$ |
| $x$ | = Input tensor x |
| $LN$ | = Layer Normalization |
| $ReLU$ | = Rectifier Linear Unit defined as $Relu(x) = max(x, 0)$ |
| $SepConv$ | = SepConv block as defined below. |

SepConv: This is depth-wise separable convolution network [1] defined as :

$$SepConv_{d,s,f}(W^{h \times w}, x) \tag{2}$$

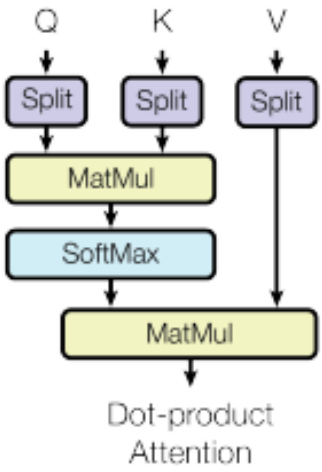


Figure 7: Dot-Product Attention Architecture. [4]

with weights W, f kernels of size $h \times w$, applied to a tensor $x$ with dimension $[batchsize, sequencelength, featurechannels]$, dialed by factor d and have stride f.
ConvBlock: A ConvBlock component consist of combination of different ConvSteps components. It is actually a combination of multiple operation back to back. These operations are defined as following:

$$hidden1(x) = ConvStep(W^{(}3 \times 1)_{h1}, x) \tag{3}$$

$$hidden2(x) = x + ConvStep(W^{(}3 \times 1)_{h2}, hidden1(x)) \tag{4}$$

$$hidden3(x) = ConvStep(W^{(}15 \times 1)_{h3}, hidden2(x)) \tag{5}$$

$$hidden4(x) = x + ConvStep(W^{(}3 \times 1)_{h4}, hidden3(x)) \tag{6}$$

$$ConvBlock(x) = Dropout(hidden4(X), 0.4)DuringTraining \tag{7}$$

$$ConvBlock(x) = hidden4(X)Else \tag{8}$$

Mixture of Experts (MoE): A mixture of experts [5] can be defined as a number of feed-forward neural networks and a trainable gating network which selects a sparse combination of the experts to process each input. For multi-model architecture 240 models were used. A graphical representation of Mixture of Experts is shown below:

## 2.4 Modality Nets

Modlity nets are the modules which conditions any specific type of input into numbers and at the output side it converts numbers back to the required format. There are four different type of input which can be handled by the multi-model architecture and these types are Language, Image, Categorical and Audio. Each modality net is named against its type.

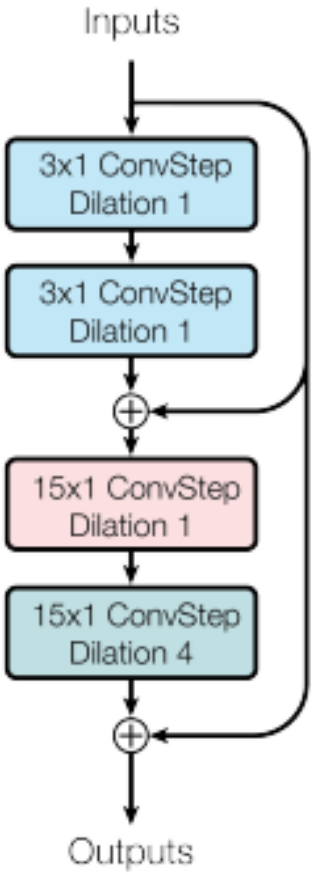


Figure 8: ConvBlock Architecture. [4]

### 2.4.1 Language Modality Net

When input is a kind of natural language, input sentence is first tokenized with the vocabulary of eight thousand words. These tokenized vectors are then transformed using learned embeddings.
When output of natural language type is needed, output vector is mapped with linear mapping and passed through softmax function to extract required words.

### 2.4.2 Image Modality Net

Image modality is composed of multiple Residual Convolution Blocks(ConvRes). Each ConvRes is composed of multiple ConvStep blocks. Mathematically a ConvRes block can be defined as ;

$$c1(x, F) = ConvStep_{t=F}(W^3, x) \tag{9}$$

$$c2(x, F) = ConvStep_{t=F}(W^3, c1(x, F)) \tag{10}$$

$$p1(x, F) = MaxPool_2([3 \times 3], c2(x, F)) \tag{11}$$

$$ConvRes(x, F) = p1(x, F) + ConvStep_{s=2}(W^{1\times1}, x) \tag{12}$$

Now that ConvRes is defined, Image Modality Net can be defined as:

$$h1(x) = ConvStep_{s=2,f=32}(W^{3\times3}, x) \tag{13}$$

$$h2(x) = ConvStep_{f=64}(W^{3\times3}, h1(x)) \tag{14}$$

$$r1(x) = ConvRes(h2(x), 128) \tag{15}$$

$$r2(x) = ConvRes(r1(x), 256) \tag{16}$$

$$ImageModality_{in}(x) = ConvRes(r2(x), d) \tag{17}$$

### 2.4.3 Categorical Modality Net

Categorical modality net takes in categorical values type input and converts it to a number representation. Categorical modality net can be defined as:

$$skip(x) = ConvStep_{s=2}({W^{3\times3}}_{skip}, x) \tag{18}$$

$$h1(x) = ConvStep({W^{3\times3}}_{h1}, x) \tag{19}$$

$$h2(x) = ConvStep({W^{3\times3}}_{h2}, x) \tag{20}$$

$$h3(x) = skip(x) + MaxPool_2([3 \times 3], h2(x)) \tag{21}$$

$$h4(x) = ConvStep_{f=1536}({W^{3\times3}}_h4, h3(x)) \tag{22}$$

$$h5(x) = ConvStep_{f=2048}(W^{3\times3}, h4(x)) \tag{23}$$

$$h6(x) = GlobalAvgPool(ReLU(h5(x))) \tag{24}$$

$$CategoricalModelity_{out}(x) = PointwiseConv(w^{classes}, h6(x)) \tag{25}$$

### 2.4.4 Audio Modality Net

Audio modality net is also composed of Residual Convolution Blocks(ConvRes). There are eight ConvRes blocks in the audio modality net concatenated back to back. A $L_{ith}$ ConvRes block can be defined as $ConvRes(Li, 2i)$

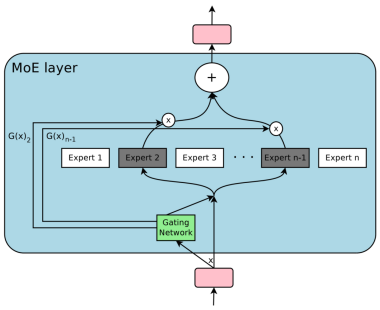


Figure 9: Mixture of Experts (MoE) Architecture. [4]

# 3 Experiments

All experiments were using tensorflow in a large distributed network. Also for all experiments Adam Optimizer optimization function along with same hyperparameter setting were used.

## 3.1 Datasets

Since there were eight tasks to be solved with multi-model architecture, eight different datasets were used which are following:

- WSJ speech corpus (Audio)
- ImageNet dataset (Image Classification)
- COCO image captioning dataset (Image Captioning)
- WSJ parsing dataset (NLP parsing)
- WMT English-German translation corpus (NLP tralation)
- WMT German-English translation corpus (NLP translation)
- WMT English-French translation corpus (NLP translation)
- WMT French-English translation corpus (NLP translation)

## 3.2 Basic Questions

Experiments are oriented to find answers of some critical questions. These questions are :

1. How far is the multi-model trained on 8 tasks simultaneously from state-of-the-art results?
2. How does training on 8 tasks simultaneously compare to training on each task separately?
3. how do the different computational blocks discussed above influence different tasks?

Each experiment use relevant datasets to answer relevant question. The answer of these questions or hypothesis are found in form of results.

| Problem | MultiModel (joint 8-problem) | State of the art |
|---|---|---|
| ImageNet (top-5 accuracy) | 86% | 95% |
| WMT EN → DE (BLEU) | 21.2 | 26.0 |
| WMT EN → FR (BLEU) | 30.5 | 40.5 |

Figure 10: Comparing MultiModel to state-of-the-art from [**?**] and [5]

### 3.3 Results

When multi-model is trained to categorize images using ImageNet, to translate English sentences to German and to translate English sentences to French, performance is not far from the state-of-the-art. As shown in figure 10, for each task, performance is close to state-of-the-art.

Similarly, to measure the difference of performance of multi-model architecture when trained together with other tasks and trained independently, few tasks are first trained together and later independently. The tasks are: Image categorization using ImageNet dataset, English to German sentence translation, Speech recognition using WSJ speech corpus and sentence parsing. Results, as shown in figure 11, show that in general simultaneous training help in performance especially when the dataset of a task is of small size. One possible explanation is internal transfer learning from other tasks.

| Problem | Joint 8-problem | | Single problem | |
|---|---|---|---|---|
| | log(perpexity) | accuracy | log(perplexity) | accuracy |
| ImageNet | 1.7 | 66% | 1.6 | 67% |
| WMT EN→DE | 1.4 | 72% | 1.4 | 71% |
| WSJ speech | 4.4 | 41% | 5.7 | 23% |
| Parsing | 0.15 | 98% | 0.2 | 97% |

Figure 11: Comparison of the MultiModel trained jointly on 8 tasks and separately on each task

Another experiment is training sentence parsing task with ImageNet image categorization task and the performance is compared with the models trained alone and with all eight tasks. It can be see that by training with ImageNet, even though it is unrelated with the parsing

task, the performance is improved. The basic assumption is the same as above that latent transfer learning helps perform the model better.

| Problem | Alone | | | W/ ImageNet | | | W/ 8 Problems | | |
|---|---|---|---|---|---|---|---|---|---|
| | log(ppl) | acc. | full | log(ppl) | acc. | full | log(ppl) | acc. | full |
| Parsing | 0.20 | 97.1% | 11.7% | 0.16 | 97.5% | 12.7% | 0.15 | 97.9% | 14.5% |

Figure 12: Results on training parsing alone, with ImageNet, and with 8 other tasks. We report log-perplexity, per-token accuracy, and the percentage of fully correct parse trees.

Last experiment is set to answer the question of how do different computational blocks influence different tasks. In the experimental setup, ImageNet image categorization task and English to French translation task is trained using three different configurations separately. In first configuration, all computation blocks as described above are used. In the second, Mixture of Experts is not used and in the third one Attention component is not used. As seen from the result in the figure 13, it is obvious that removing components effect the performance, specially when the component is considered important for one specific task. For example, Attention component is considered important for natural language related tasks and it can be seen that removing attention component reduce the performance from 76 percent to 72 percent. Conversely, it can be seen that removing unrelated modules also negatively effects the performance but with not high value.

| Problem | All Blocks | | Without MoE | | Without Attention | |
|---|---|---|---|---|---|---|
| | log(perpexity) | accuracy | log(perplexity) | accuracy | log(perplexity) | accuracy |
| ImageNet | 1.6 | 67% | 1.6 | 66% | 1.6 | 67% |
| WMT EN→FR | 1.2 | 76% | 1.3 | 74% | 1.4 | 72% |

Figure 13: Ablating mixture-of-experts and attention from MultiModel training.

# 4 Conclusion

As proposed in the beginning, a model was designed to solve multiple tasks with different input and output modalities. These tasks were language translation, image categorization, image captioning and language tree parsing. Although performance for each task was not state-of-the-art but still comparable. Tasks with less dataset benefit the most. Also since experiments were performed with only single-hyperparameter set there are high chances of increase in performance after hyperparamter tuning. The multi-model architecture training took one week with large distributed network.

# 5 Discussions and Remarks

## 5.1 General Review

As general review, the multi-model architecture is a kind of black-box deep learning model which is limited by specialist knowledge. Thus it will be difficult to make sense weights assigned. Positively, it was a good try to imitate human brain. Resource wise it is a good way to compress multiple models and decrease training time and storage resources.

## 5.2 Technical Review

As technical review, the general approach of multi-model architecture is modular which can be used as a motivation to build other similar architectures and improve on it. Overall multi-model architecture tires to have a module for each input type and a specialized network for each special task which are connected through a control. This control is learned in form of weights and gating system. Experiments were performed well to answer the questions although the quantity is way less to generalize the results.

# 6 References

‘


[1] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *CoRR*, vol. abs/1610.02357, 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[2] L. et. al. (2017) Author discussion. [Online]. Available: https://github.com/tensorflow/tensor2tensor/issues/9

[3] ——. (2017) Github Code github code. [Online]. Available: https://github.com/tensorflow/tensor2tensor/blob/f9c859a7639831c6da864e360793a285613dcc5c/tensor2tensor/models/multimodel.py

[4] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, “One model to learn them all,” *CoRR*, vol. abs/1706.05137, 2017. [Online]. Available: http://arxiv.org/abs/1706.05137

[5] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” *CoRR*, vol. abs/1701.06538, 2017. [Online]. Available: http://arxiv.org/abs/1701.06538