Introduction to Neural Machine Translation

|  |  |
| --- | --- |
| **Summary** | In this codelab, you’ll build your first NMT system using TensorFlow and Compute Engine, view a summary of the model during training with TensorBoard. |
| **URL** |  |
| **Category** | TensorFlow |
| **Environment** |  |
| **Status** | Draft |
| **Feedback Link** |  |
| **Author** | gmikels |

1. Install [the Preview Codelab Chrome extension](https://chrome.google.com/webstore/detail/preview-codelab/lhojjnijnkiglhkggagbapfonpdlinji) in your browser. It should create a button in your browser that looks like this → 
2. Preview your codelab by clicking the Chrome extension’s button (pictured above) while you’re in your codelab Google Doc tab. It will open a new tab to preview.

[Introduction](#_nitx1pj43rcc)

[Background on Neural Machine Translation](#_payr7g26yhol)

[What you will build](#_kym9ahd9nmy6)

[What you’ll learn](#_9ni8nt5a8kl4)

[What you’ll need](#_xplkkg3ediae)

[Getting set up](#_bwo0af2iwk90)

[Create a New Project in Google Cloud Console](#_uvmd9z37yom4)

[Create a VM using Compute Engine for Training & Inference](#_ww8gynruj865)

[Create a Firewall Rule for TensorBoard](#_5qj8ndq35q7t)

[Execute an NMT Training Job on Compute Engine](#_ot43tirxvs6w)

[SSH into Linux VM](#_sflrq0mndtt9)

[Install Dependencies](#_5trlpk27coqk)

[Download Source Code from Github](#_dw9xhaeh968d)

[Execute Training](#_bzcfze2ys9yy)

[View Training Progress with Tensorboard](#_beqyjnbcvpfv)

[Open a 2nd SSH Window](#_qfbk0dtvpf2a)

[Start Tensorboard](#_ts76b9ir2u51)

[Open Tensorboard in a Browser](#_styxhoo3i8ur)

[Generate Inferences from the Trained Model](#_atck803uyi0q)

[Copy Checkpoints from a Previous Training Job](#_7nf0k6ptljkc)

[Create a File with Sentences to Translate](#_id8xgyoxo2aw)

[Execute a Script to Generate English Translations](#_hj1l509pcyzx)

[Compare Your Results with the Translation API](#_6gjpd64peceq)

# Introduction

Sequence-to-sequence (seq2seq) models ([Sutskever et al., 2014](https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf), [Cho et al., 2014](http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf)) have enjoyed great success in a variety of tasks such as machine translation, speech recognition, and text summarization. This tutorial will focus on the task of Neural Machine Translation (NMT) which was the very first testbed for seq2seq models with wild [success](https://research.googleblog.com/2016/09/a-neural-network-for-machine.html). The included code is lightweight, high-quality, production-ready, and incorporated with the latest research ideas.

# Background on Neural Machine Translation

Back in the old days, traditional phrase-based translation systems performed their task by breaking up source sentences into multiple chunks and then translated them phrase-by-phrase. This led to disfluency in the translation outputs and was not quite like how we, humans, translate. We read the entire source sentence, understand its meaning, and then produce a translation. Neural Machine Translation (NMT) mimics that!

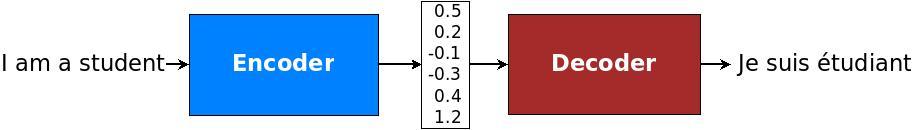


Figure 1. **Encoder-decoder architecture** – example of a general approach for NMT. An encoder converts a source sentence into a "meaning" vector which is passed through a *decoder* to produce a translation.

Specifically, an NMT system first reads the source sentence using an *encoder* to build a ["thought" vector](https://www.theguardian.com/science/2015/may/21/google-a-step-closer-to-developing-machines-with-human-like-intelligence), a sequence of numbers that represents the sentence meaning; a *decoder*, then, processes the sentence vector to emit a translation, as illustrated in Figure 1. This is often referred to as the *encoder-decoder architecture*. In this manner, NMT addresses the local translation problem in the traditional phrase-based approach: it can capture *long-range dependencies* in languages, e.g., gender agreements; syntax structures; etc., and produce much more fluent translations as demonstrated by [Google Neural Machine Translation systems](https://research.googleblog.com/2016/09/a-neural-network-for-machine.html).

NMT models vary in terms of their exact architectures. A natural choice for sequential data is the recurrent neural network (RNN), used by most NMT models. Usually an RNN is used for both the encoder and decoder. The RNN models, however, differ in terms of: (a) *directionality* – unidirectional or bidirectional; (b) *depth* – single- or multi-layer; and (c) *type*– often either a vanilla RNN, a Long Short-term Memory (LSTM), or a gated recurrent unit (GRU). Interested readers can find more information about RNNs and LSTM on this [blog post](http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

In this tutorial, we consider as examples a *deep multi-layer RNN* which is unidirectional and uses LSTM as a recurrent unit. We show an example of such a model in Figure 2. In this example, we build a model to translate a source sentence "I am a student" into a target sentence "Je suis étudiant". At a high level, the NMT model consists of two recurrent neural networks: the *encoder* RNN simply consumes the input source words without making any prediction; the *decoder*, on the other hand, processes the target sentence while predicting the next words.

For more information, we refer readers to [Luong (2016)](https://github.com/lmthang/thesis) which this tutorial is based on.

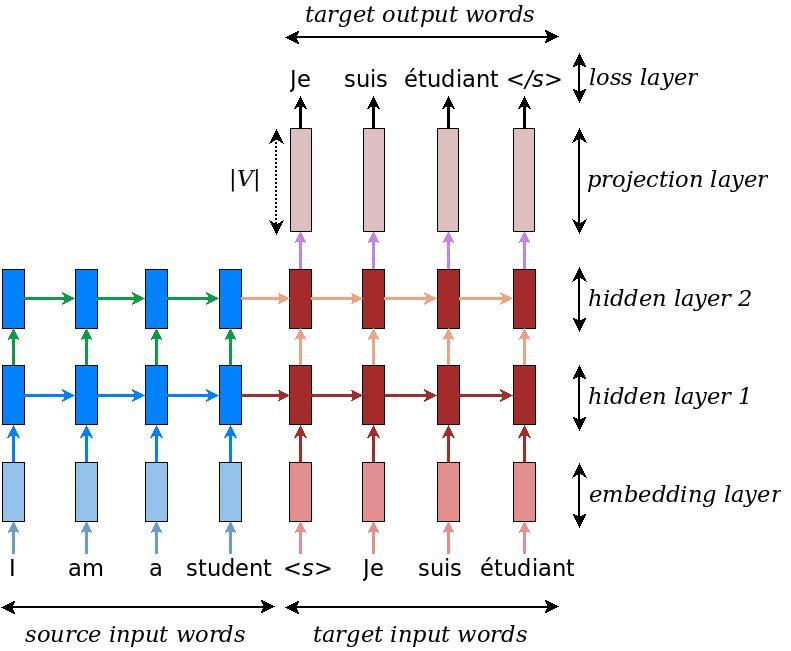


Figure 2. **Neural machine translation** – example of a deep recurrent architecture proposed by for translating a source sentence "I am a student" into a target sentence "Je suis étudiant". Here, "<s>" marks the start of the decoding process while "</s>" tells the decoder to stop.

## What you will build

In this codelab, you’ll build your first NMT system using TensorFlow and Compute Engine and view a summary of the model during training with TensorBoard. You will:

* Use a *small-scale parallel corpus of TED talks* (133K training examples)
* Train your very first NMT model, translating from Vietnamese to English

## What you’ll learn

* How to train a simple NMT model
* How to generate inferences from a trained seq2seq model

## What you’ll need

* A recent version of [Chrome](https://www.google.com/chrome/). Note, this works in other browsers as well, but we’ll be using a few features of the Chrome DevTools to better understand what’s happening at the browser level.
* Basic knowledge of Google Cloud Shell and Linux command line tools.

This codelab is focused on a simple approach to Neural Machine Translation. Non-relevant code blocks are glossed over and are provided for you to simply copy and paste.

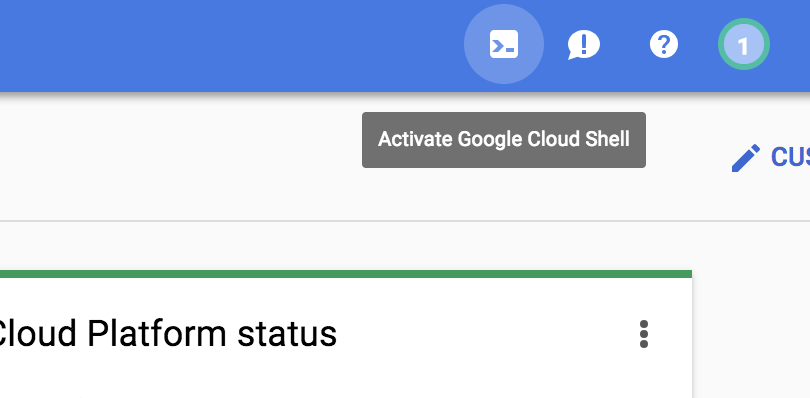
# Getting set up

## Create a New Project in Google Cloud Console

Click the following link to open the Google Cloud Console:

[Open Google Cloud Console](https://console.cloud.google.com)

Once created, open your new project and activate Cloud Shell.



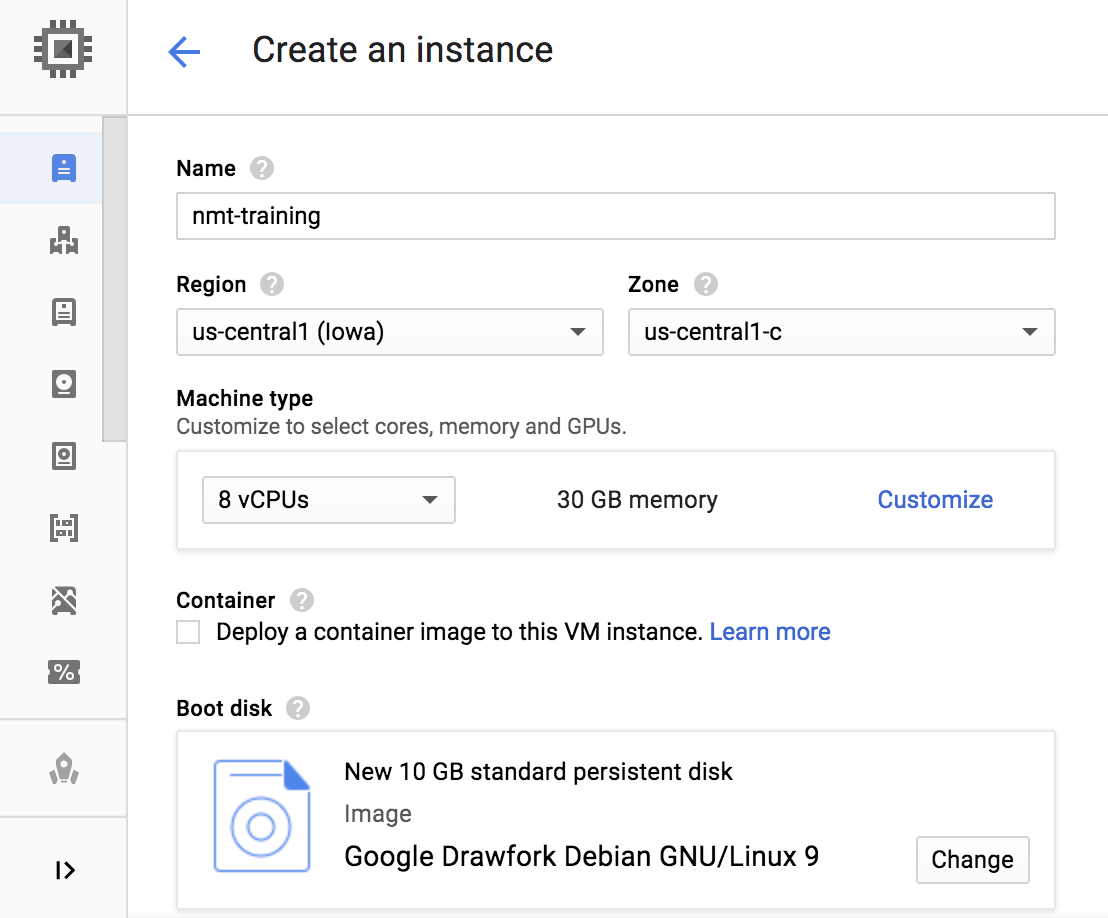
## Create a VM using Compute Engine for Training & Inference

In this step, you will use Cloud Shell to create a Linux VM that will be used for training and testing the NMT model.

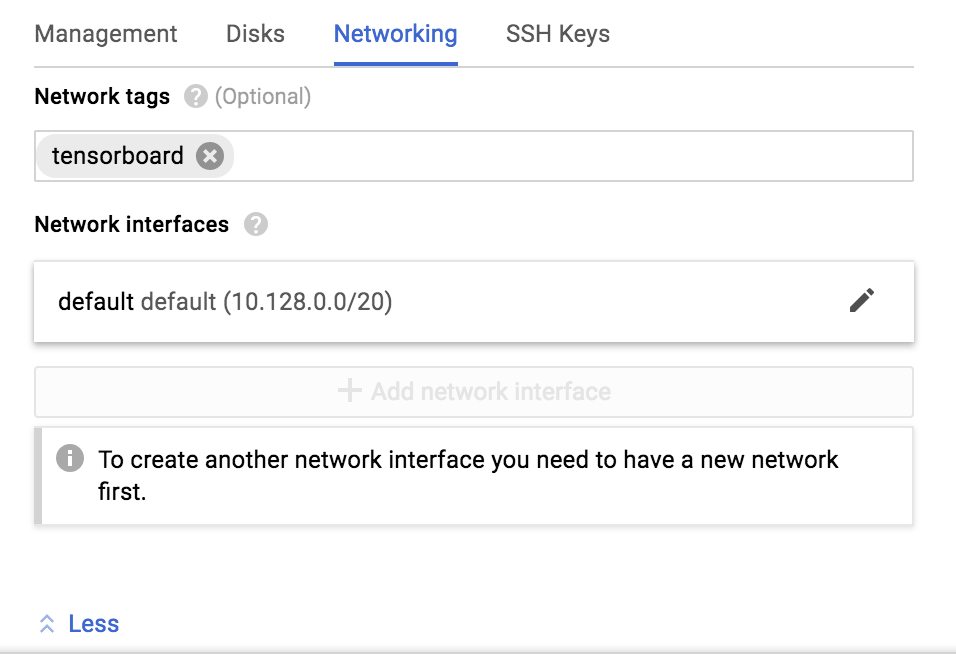
Navigate to Compute Engine > VM Instances. Select “Create” and configure the following:

Name: nmt-training

Machine type: n1-standard-8



Select “Management, disks, networking, SSH keys” and click on the “Networking” tab. Add “tensorboard” as a network label.

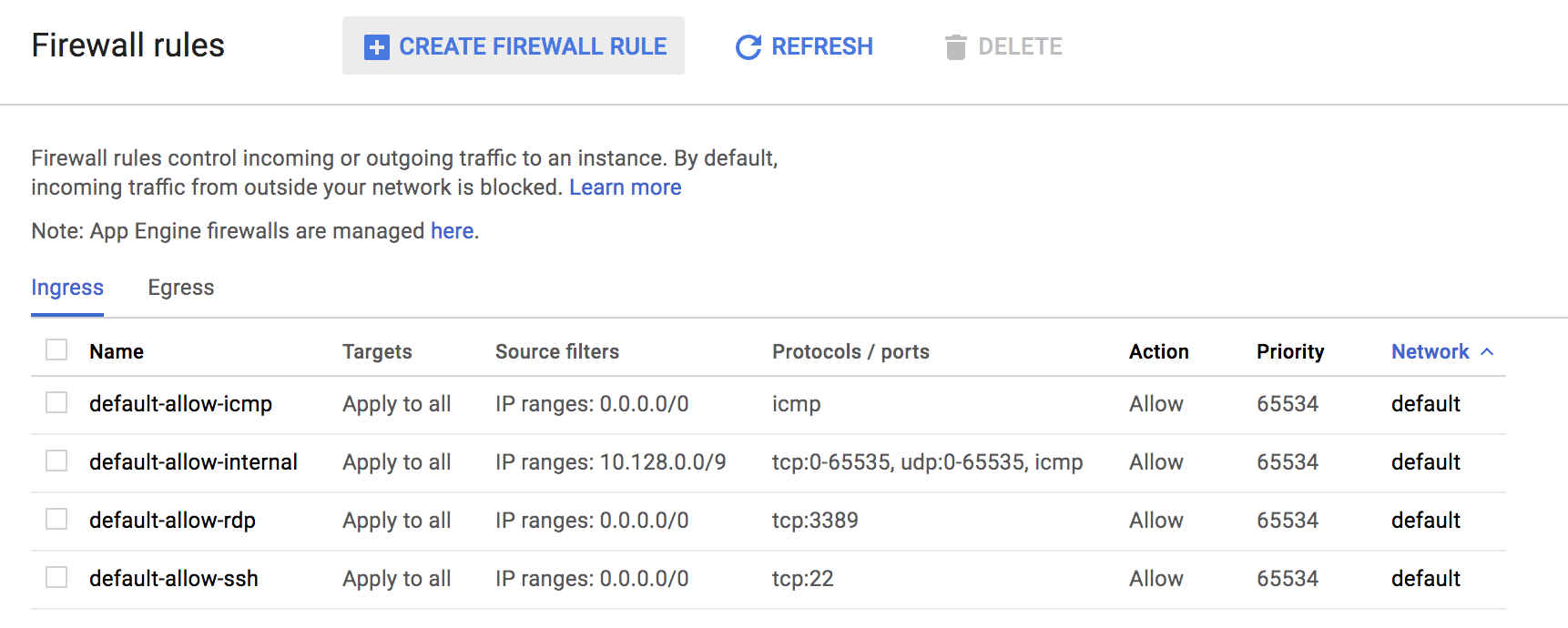


Leave the defaults in the remaining fields and click “Create” when finished.

## Create a Firewall Rule for TensorBoard

In this step you will create a firewall rule that will allow you to connect to TensorBoard on the VM you just created.

Navigate to VPC Network > Firewall Rules. Click “Create Firewall Rule”.



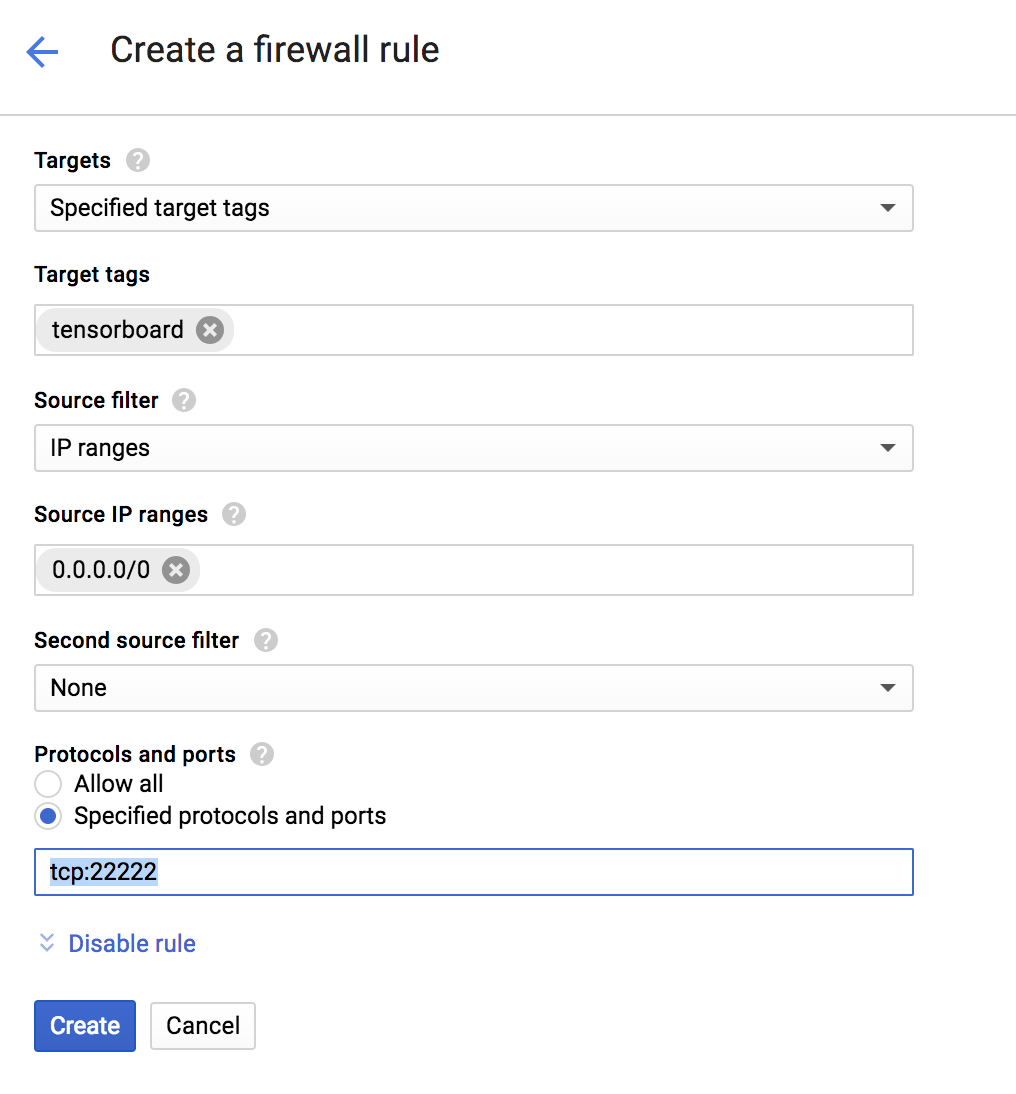
Populate the following fields and select “Create”:

Name: tensorboard

Target tags: tensorboard

Source IP ranges: 0.0.0.0/0

Specified ports and protocols: tcp:22222

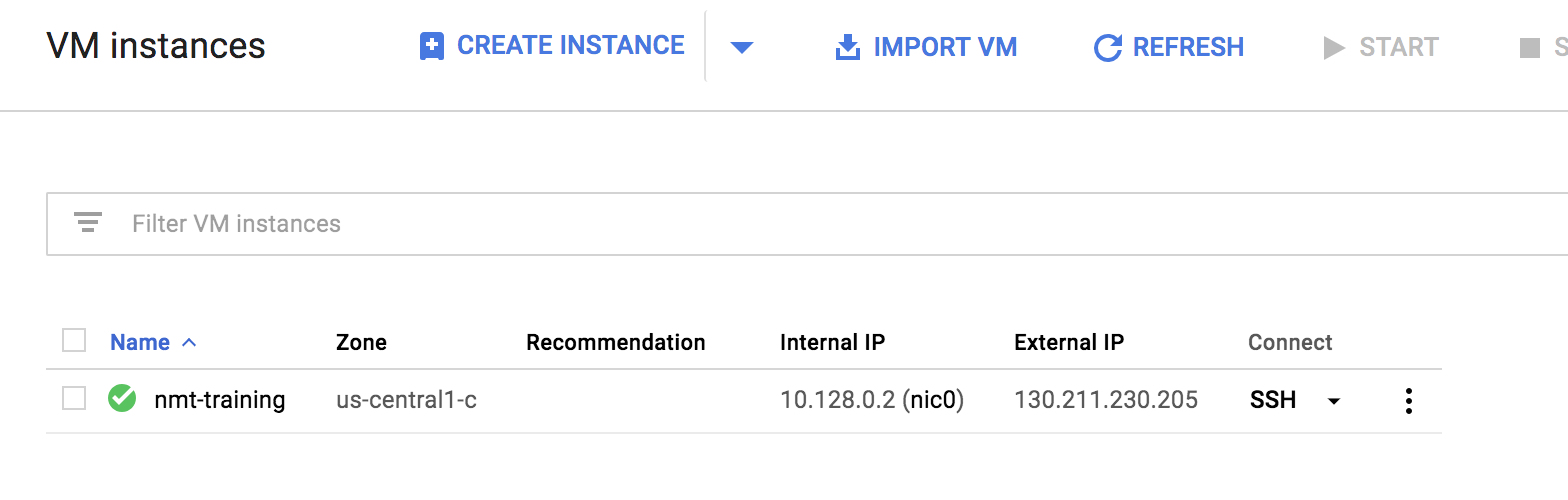


Click on Create.

# Execute an NMT Training Job on Compute Engine

## SSH into Linux VM

Navigate to Compute Engine > VM Instances. Select “SSH” to connect to the “nmt-training” VM.



## Install Dependencies

Run the following in the Linux shell to install dependencies:

|  |
| --- |
| sudo apt update  wget https://bootstrap.pypa.io/get-pip.py sudo python get-pip.py sudo apt install git-all --assume-yes  sudo pip install tensorflow |

## Download Source Code from Github

Run the following to download the source code for this example from Github:

|  |
| --- |
| git clone https://github.com/tensorflow/nmt.git |

## Execute Training

Run the following to download a small set of training data (*small-scale parallel corpus of TED talks* -- 133K training examples) and create a directory to store the trained model.

|  |
| --- |
| cd nmt nmt/scripts/download\_iwslt15.sh ./nmt\_data mkdir ./nmt\_model |

Run the following to execute training. The entry point for training is the nmt.py script. This command will execute a training job with 12000 steps, with checkpoints created every 1000 steps.

|  |
| --- |
| python -m nmt.nmt \  --src=vi --tgt=en \  --vocab\_prefix=nmt\_data/vocab \  --train\_prefix=nmt\_data/train \  --dev\_prefix=nmt\_data/tst2012 \  --test\_prefix=nmt\_data/tst2013 \  --out\_dir=nmt\_model \  --num\_train\_steps=12000 \  --steps\_per\_stats=100 \  --num\_layers=2 \  --num\_units=128 \  --dropout=0.2 \  --metrics=bleu |

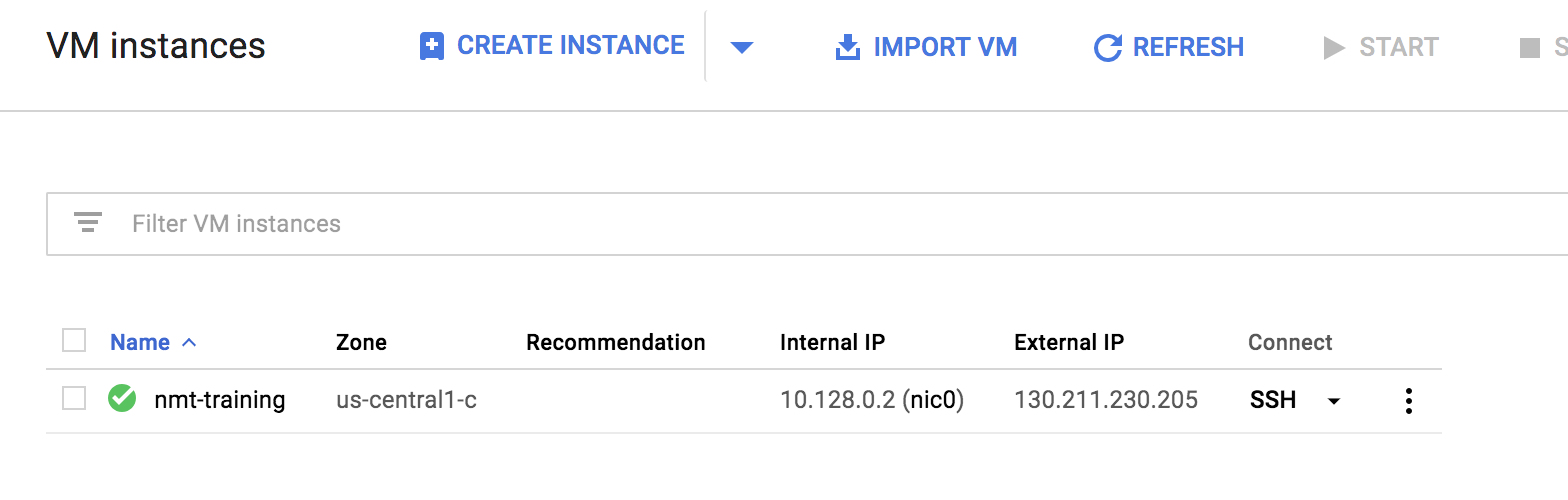
While this job is running, continue to the next step to view training progress in Tensorboard. For the sake of time, **we will cancel this job and download model checkpoints** in a subsequent section.

# View Training Progress with Tensorboard

## Open a 2nd SSH Window

Navigate to Compute Engine > VM Instances. Select “SSH” to open a 2nd shell window for the “nmt-training” VM.

Make a note of the External IP of the VM to be used in the next step.



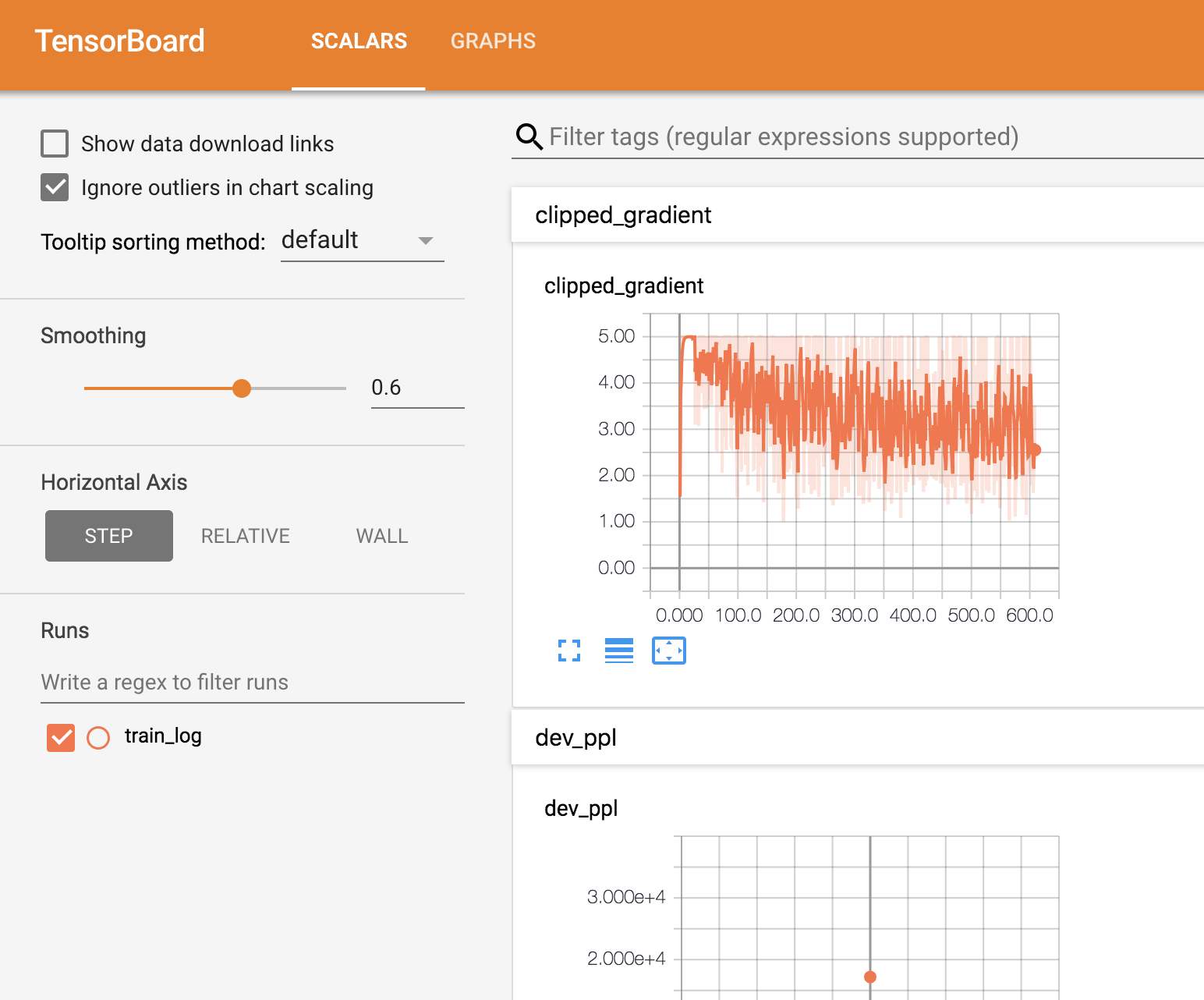
## Start Tensorboard

Run the following to start an instance of Tensorboard:

|  |
| --- |
| tensorboard --port 22222 --logdir nmt/nmt\_model/ |

## Open Tensorboard in a Browser

Using the external IP address of your VM, type “[external\_IP]:22222” in the navigation bar of your browser.



## 

# Generate Inferences from the Trained Model

## Copy Checkpoints from a Previous Training Job

Type “CTRL-C” to cancel the currently running training script. Run the following commands to download checkpoints from a previous training job:

|  |
| --- |
| git clone <https://github.com/gmikels/model_checkpoints.git> \ ~/nmt  cp ~/nmt/model\_checkpoints/\* ~/nmt/nmt\_model/ |

## Create a File with Sentences to Translate

Here is a list of Vietnemese sentences that can be added to the input file for translation:

|  |
| --- |
| Tôi đã rất tự hào về đất nước tôi . Ở trường , chúng tôi dành rất nhiều thời gian để học về cuộc đời của chủ tịch Kim II- Sung , nhưng lại không học nhiều về thế giới bên ngoài , ngoại trừ việc Hoa Kỳ , Hàn Quốc và Nhật Bản là kẻ thù của chúng tôi . Mặc dù tôi đã từng tự hỏi không biết thế giới bên ngoài kia như thế nào , nhưng tôi vẫn nghĩ rằng mình sẽ sống cả cuộc đời ở BắcTriều Tiên , cho tới khi tất cả mọi thứ đột nhiên thay đổi . Khi tôi lên 7 , tôi chứng kiến cảnh người ta xử bắn công khai lần đầu tiên trong đời , nhưng tôi vẫn nghĩ cuộc sống của mình ở đây là hoàn toàn bình thường . Gia đình của tôi không nghèo , và bản thân tôi thì chưa từng phải chịu đói . Nhưng vào một ngày của năm 1995 , mẹ tôi mang về nhà một lá thư từ một người chị em cùng chỗ làm với mẹ . |

Run the following command to create a file with examples to translate, copying the sentences above. Type “CTRL-C” after the copying the sentences into the shell.

|  |
| --- |
| cat > ./my\_infer\_file.vi # (copy and paste some sentences from above)  # type ctrl-c when finished |

## Execute a Script to Generate English Translations

Run the command below to translate the sentences you copied above from Vietnemese to English. The output file will be generated in ~/nmt/nmt\_model.

|  |
| --- |
| cd ~/nmt  python -m nmt.nmt \  --out\_dir=nmt\_model \  --inference\_input\_file=my\_infer\_file.vi \  --inference\_output\_file=nmt\_model/output\_infer |

Run the following to view the translated output:

|  |
| --- |
| cat nmt\_model/output\_infer |

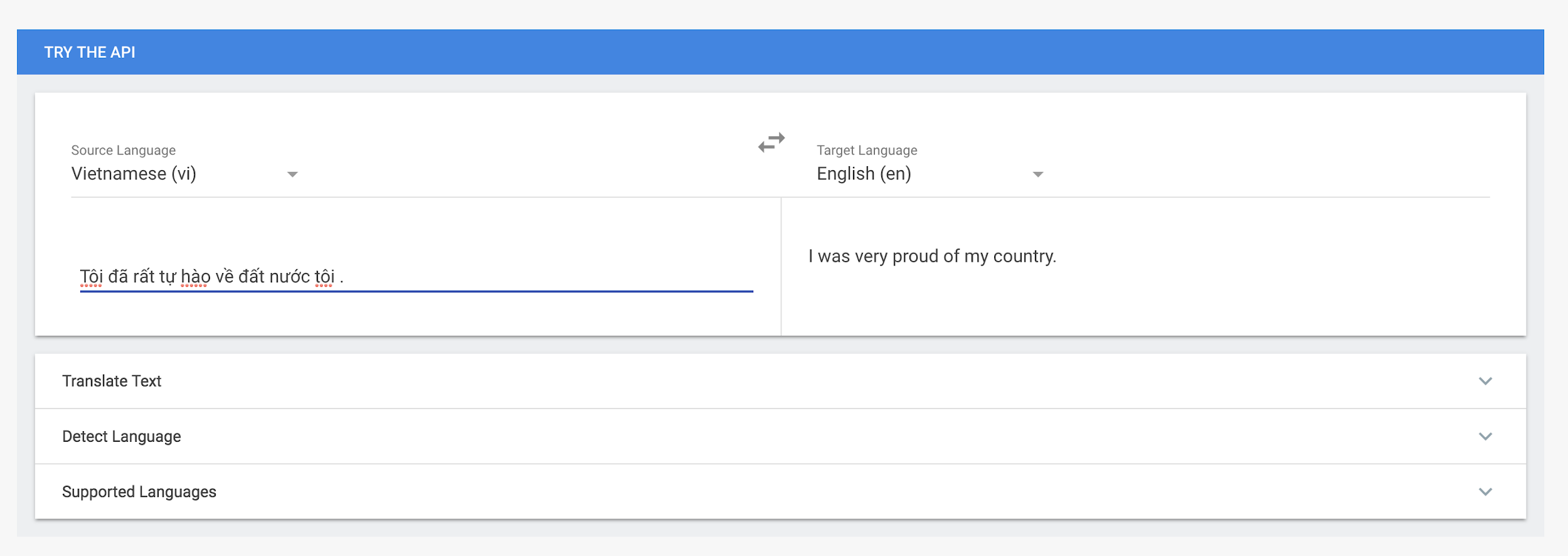
## Compare Your Results with the Translation API

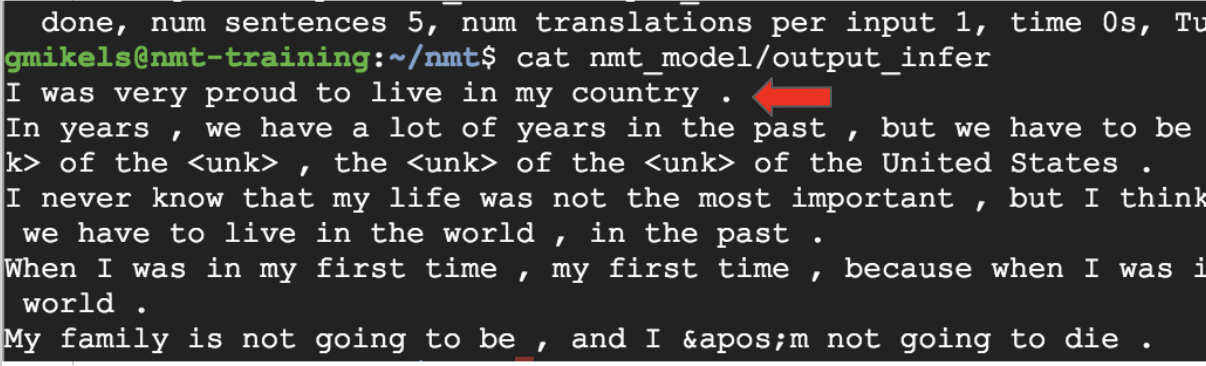
Navigate to the link below to try the Translation API.

[Try the Translation API](https://cloud.google.com/translate/)

Our model performed well with this sentence:

|  |
| --- |
| Tôi đã rất tự hào về đất nước tôi . |





But not as well with this one:

|  |
| --- |
| Ở trường , chúng tôi dành rất nhiều thời gian để học về cuộc đời của chủ tịch Kim II- Sung , nhưng lại không học nhiều về thế giới bên ngoài , ngoại trừ việc Hoa Kỳ , Hàn Quốc và Nhật Bản là kẻ thù của chúng tôi . |

