Kivy: Deep Recipe Embeddings for Generation, Retrieval, and Nutritional Analysis

1 Abstract

Understanding and representing culinary recipes in a meaningful and machine-interpretable way is a growing area of interest in food computing, with applications in recommendation systems, dietary planning, and generative cooking models. In this paper, we present two major contributions to the field. First, we release a newly curated and cleaned dataset of over 500k recipes scraped from Food.com, containing structured ingredient lists, cooking instructions, titles, and metadata such as cuisine and dietary tags. This dataset offers a high-quality, diverse, and reproducible benchmark for future research. Second, we propose a novel autoencoder-based architecture designed to learn robust recipe embeddings that capture semantic, structural, and functional aspects of recipes. By training the model to reconstruct full recipes from latent representations, the autoencoder learns embeddings that are compact yet informative. We demonstrate the effectiveness of these embeddings across several downstream tasks, including ingredient-based recipe retrieval, cuisine classification, and recipe similarity search, achieving competitive results without relying on multimodal inputs. Our findings suggest that autoencoder-based embeddings can serve as a strong foundation for a wide range of intelligent culinary applications, and we provide our code and dataset publicly to support further exploration in this domain.

2 Introduction

2.1 Motivation

Food is a central aspect of human life, impacting not only health and well-being but also culture, identity, and sustainability. As digital platforms increasingly become the primary source of culinary information, there is a growing need to build intelligent systems that can understand, organize, and personalize recipe content. However, recipes are inherently complex: they combine structured and unstructured data, rely heavily on domain-specific language, and often lack standardization. Learning meaningful representationsor embeddingsof recipes can enable machines to reason about food in a way that is both flexible and scalable.

Such embeddings have wide-ranging applications: they can power personalized recipe recommendation engines, facilitate ingredient-based search, support dietary and allergy-aware meal planning, or even serve as the foundation for generative models that create new dishes. Despite these possibilities, existing recipe datasets are often noisy, outdated, or lack the granularity required for fine-grained modeling. Furthermore, current embedding methods typically rely on large multimodal setups, which may be impractical for lightweight or text-only use cases.

To address these challenges, we introduce a new curated dataset from Food.com and propose an autoencoder-based architecture that learns latent recipe representations from purely textual inputs. Our goal is to provide both high-quality data and a flexible, efficient model architecture that can serve as a foundation for the next generation of food-aware applications.

2.2 Problem Statement

Despite the growing interest in computational food understanding, there remains a lack of high-quality, structured datasets and efficient, general-purpose models for learning robust representations of textual recipe data. Existing approaches to recipe embedding often depend on multimodal data (e.g.,

images, user interactions) or rely on shallow textual features that fail to capture the deeper semantic relationships between ingredients, instructions, and culinary intent.

Moreover, many available datasets contain noisy or incomplete information, such as improperly formatted ingredient lists, ambiguous instructions, or inconsistent metadata. This significantly limits the ability of models to generalize across tasks such as retrieval, classification, or recipe generation. Without clean data and strong embeddings, intelligent food systems are constrained in their ability to offer accurate recommendations, enable creative substitutions, or support health-aware personalization.

To bridge this gap, there is a need for (1) a reliable, well-structured textual recipe dataset, and (2) a model capable of encoding recipes into dense, meaningful representations that capture their functional, cultural, and nutritional context. This paper addresses both challenges.

3 Related Work

Research on computational modeling of culinary data has expanded rapidly in recent years, driven by the emergence of large-scale recipe datasets and the increasing interest in personalized nutrition, food recommendation systems, and generative cooking tools. Much of the early work in this domain focused on rule-based or retrieval-based methods, which relied on keyword matching, user ratings, or ingredient overlaps to suggest recipes. While simple, these approaches often failed to generalize or capture semantic nuances in recipe text.

A significant advancement came with the Recipe1M dataset introduced by Salvador et al. (2017), which enabled the development of joint multimodal embeddings linking recipes and images. Their im2recipe model used a deep neural network trained with a cosine similarity loss to align visual and textual modalities in a shared embedding space. Subsequent works extended this idea, using contrastive learning (e.g., triplet loss) and transformer-based encoders to further enhance performance in cross-modal retrieval tasks.

Other studies explored sequence modeling approaches for ingredient prediction or instruction generation, often leveraging recurrent neural networks or transformer architectures. For example, Kusupati et al. (2020) used transformers to model ingredient co-occurrence, while Wang et al. (2019) proposed hierarchical models to separately embed ingredients and instructions before merging them into a unified representation.

More recently, research has focused on personalized or health-aware embeddings, where models are fine-tuned based on user preferences, dietary goals, or cultural constraints. These systems often require access to user-specific data or additional metadata, making them harder to generalize without substantial preprocessing or user feedback loops.

In contrast to prior work, our method proposes a text-only, autoencoder-based architecture that learns holistic recipe embeddings by reconstructing the full recipe from a latent space. This approach avoids reliance on image data or user interactions, offering a lightweight, interpretable, and general-purpose embedding suitable for multiple downstream tasks. Additionally, we address a major limitation in existing datasets by releasing a new, curated Food.com dataset with cleaned ingredient and instruction fields, enabling more effective model training and evaluation.

4 Dataset

To support the development of robust recipe embeddings, we introduce a curated and preprocessed dataset consisting of approximately 500,000 recipes collected from Food.com. This dataset combines multiple scrapes of the website and includes structured data spanning textual descriptions, categorical labels, nutritional facts, and detailed ingredient information.

4.1 Dataset Structure

Each recipe in the dataset contains the following attributes:

- **Textual Information**:
  - `Name` the recipe title.
  - `Description` a brief textual overview or background of the recipe.

  - `Steps` an ordered list of instructions for preparation.

- **Categorical Fields**:
  - `Tags` such as "vegan", "quick", "holiday", etc.
  - `General Category` e.g., "dessert", "main course", "appetizer".

- **Nutritional Values**:
  - `Calories`, `Cholesterol`, `Sodium`, `Carbohydrates`, `Fiber`, `Sugar`, `Protein`, `Total Fat`, `Saturated Fat`.

- **Ingredients**:
  - Each ingredient entry includes:
    * `Name`
    * `Quantity`
    * `Unit of measure`
    * `Optional metadata` (e.g., preparation style or additional descriptors)

- **Additional Metadata**:
  - `Preparation Time`, `Cooking Time`, `Serving Size`, `Author`, and `Images`.

4.2 Preprocessing and Filtering

To reduce sparsity and improve semantic consistency, we filtered out all tags, categories, and ingredients that appeared fewer than five times across the corpus. This resulted in a significant reduction of ingredient types from over 450,000 to approximately 20,000 standardized ingredients. To handle removed or rare ingredients, we applied semantic similarity-based imputation using pretrained language models, replacing rare items with the most similar frequent equivalents. The same strategy was applied to rare tags and categories.

For modeling purposes, we retained a subset of the most semantically informative features: `Name`, `Steps`, `Ingredients`, `Nutritional Information`, `Tags`, and `General Category`.

4.3 Extended Recipes via RecipeNLG

To further scale our dataset, we leveraged the RecipeNLG dataset, which contains over 2 million recipes, including many from Food.com. RecipeNLG provides `Title`, `Ingredients`, and `Steps`, but lacks nutritional data, categories, and tags. To address this, we trained a classification model using our curated 500k dataset to impute missing tags and general categories in RecipeNLG. This enables the integration of additional recipe data for training and evaluation while maintaining semantic richness.

4.4 Ingredient Nutrition Matching

We also incorporated nutritional values for individual ingredients using the USDA FoodData Central database. Ingredient-level nutrition information was aggregated per recipe and cross-checked with available recipe-level nutrition estimates when possible, increasing the reliability of health-related attributes in the dataset.

5 Methodology

5.1 Embedding architecture

Encoder Blocks for Ingredient Representation

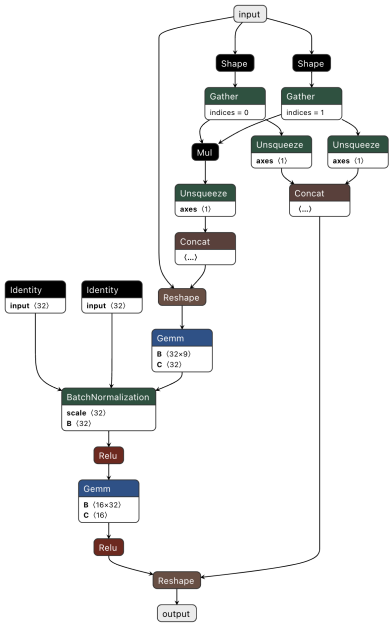


Figure 1: Nutriment encoder

1. Nutriment Block

Given a per-ingredient nutriment vector $x_n \in R^{d_n}$ $(d_n = 9)$ and a scalar quantity $q \in R$, we concatenate them to form $x_{nut} = [x_n; q] \in R^{d_n+1}$. This is processed by a feed-forward neural network with batch normalization and dropout:

$$h_1 = ReLU(BN(W_1 x_{nut} + b_1))$$

$$h_1' = Dropout(h_1)$$

$$h_2 = ReLU(W_2 h_1' + b_2)$$

where W_1, W_2 are weight matrices, b_1, b_2 are biases, and BN is batch normalization.
We will define this set of operations as the **NutEnc** function:

$$h = NutEnc(x)$$

2. Name Encoder Block (BERT)

Given an ingredient name as a sequence of tokens $\mathbf{x}_{name} = (w_1, w_2, ..., w_L)$, encode it with a pretrained BERT model:

$$\mathbf{e}_{name} = \mathbf{BERT}_{[\text{CLS}]}(\mathbf{x}_{name}) \in R^{d_{bert}}$$

where $\mathbf{BERT}_{[\text{CLS}]}$ denotes the output of the [CLS] token. Then, project to a lower dimension:

$$\mathbf{e}_{name}^{proj} = \mathbf{W}_p \mathbf{e}_{name} + \mathbf{b}_p$$

3. Fusion Block

Fuse the numeric ingredient representation $\mathbf{h}_2$ and BERT name encoding $\mathbf{e}_{name}^{proj}$ by concatenation:

$$\mathbf{z}_{fuse} = [\mathbf{h}_2; \mathbf{e}_{name}^{proj}]$$

Pass through another MLP with batch normalization and dropout:

$$\mathbf{f}_1 = ReLU(BN_f(\mathbf{W}_f\mathbf{z}_{fuse} + \mathbf{b}_f))$$

$$\mathbf{f}'_1 = Dropout(\mathbf{f}_1)$$

$$\mathbf{z}_{ingredient} = ReLU(\mathbf{W}_o\mathbf{f}'_1 + \mathbf{b}_o)$$

4. Ingredient Set Aggregator (Transformer Encoder)

Given the set of N encoded ingredient vectors $\{\mathbf{z}^{(i)}_{ingredient}\}_{i=1}^{N}$, stack them as a sequence and feed into a Transformer encoder:

$$\mathbf{Z}_{trans} = TransformerEncoder\left(\left[\mathbf{z}^{(1)}_{ingredient}, \ldots, \mathbf{z}^{(N)}_{ingredient}\right]\right)$$

where $\mathbf{Z}_{trans} \in R^{N \times d_{trans}}$.

5. Set Pooling

Aggregate the sequence of transformed ingredient vectors via mean pooling:

$$\mathbf{z}_{recipe} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{Z}^{(i)}_{trans}$$

6. Recipe Name Encoder

Let the recipe title or name be a sequence of tokens $\mathbf{x}_{title} = (w_1, ..., w_T)$. Encode with BERT and optionally project:

$$\mathbf{e}_{title} = \mathbf{BERT}_{[\text{CLS}]}(\mathbf{x}_{title}) \in R^{d_{bert}}$$

$$\mathbf{z}_{title} = \mathbf{W}_t\mathbf{e}_{title} + \mathbf{b}_t$$

7. Recipe Nutriments Encoder

Let the recipe-level nutriments vector be $\mathbf{x}_{nutr,rec} \in R^{d_{nutr,rec}}$. Pass through a feed-forward network:

$$\mathbf{h}_{nutr,rec} = ReLU(BN_r(\mathbf{W}_r\mathbf{x}_{nutr,rec} + \mathbf{b}_r))$$

$$\mathbf{z}_{nutr,rec} = ReLU(\mathbf{W}'_r\mathbf{h}_{nutr,rec} + \mathbf{b}'_r)$$

8. Steps Encoder

Let the recipe steps be a sequence of tokens $\mathbf{x}_{steps} = (w_1, ..., w_S)$. Encode with BERT (or another transformer, or mean-pool the step-wise BERT [CLS] outputs):

$$\mathbf{e}_{steps} = \mathbf{BERT}_{[\text{CLS}]}(\mathbf{x}_{steps}) \in R^{d_{bert}}$$

$$\mathbf{z}_{steps} = \mathbf{W}_s\mathbf{e}_{steps} + \mathbf{b}_s$$

9. Fusion Block

Concatenate all high-level recipe representations:

$$\mathbf{z}_{all} = [\mathbf{z}_{recipe}; \mathbf{z}_{title}; \mathbf{z}_{nutr,rec}; \mathbf{z}_{steps}]$$

Pass through a fusion MLP with batch normalization and dropout:

$$\mathbf{f}_{all,1} = ReLU(BN_a(\mathbf{W}_a\mathbf{z}_{all} + \mathbf{b}_a))$$

$$\mathbf{f}'_{all,1} = Dropout(\mathbf{f}_{all,1})$$

$$\mathbf{z}_{final} = ReLU(\mathbf{W}_f\mathbf{f}'_{all,1} + \mathbf{b}_f)$$

10. Final Recipe Representation

$$\mathbf{z}_{final} = Fusion(\mathbf{z}_{recipe},\ \mathbf{z}_{title},\ \mathbf{z}_{nutr,rec},\ \mathbf{z}_{steps})$$

5.2 Training Objectives

Given the unified recipe embedding $\mathbf{z}_{final}$ (see previous sections), we propose the following multi-task training objectives to learn rich, useful representations for recipes and their components:

1. Autoencoder Objective

We employ an autoencoder to encourage the embedding to capture the essential information of the recipe. The encoder E maps the input recipe data to $\mathbf{z}_{final}$, and the decoder D attempts to reconstruct the original raw features (or a summary thereof).

Let $\mathbf{x}_{raw}$ be the raw input features (could be concatenation of ingredient/step/nutriment representations, or a lower-dimensional summary). The autoencoder loss is:

$$\mathcal{L}_{AE} = \|D(\mathbf{z}_{final}) - \mathbf{x}_{raw}\|_2^2$$

2. Nutriment Prediction Objective

We attach a regressor head R to $\mathbf{z}_{final}$ to predict recipe-level target nutriments (e.g., calories, protein, fat, etc.). Let $\mathbf{y}_{nutr,target} \in R^{d_{nutr,rec}}$ be the ground truth nutriments for the recipe. The regressor outputs $\hat{\mathbf{y}}_{nutr} = R(\mathbf{z}_{final})$.

The nutriment regression loss is:

$$\mathcal{L}_{nutr} = \|\hat{\mathbf{y}}_{nutr} - \mathbf{y}_{nutr,target}\|_2^2$$

3. Tag and Category Classification Objective

Let the recipe have categorical labels such as tags (multi-label) and/or categories (multi-class).

- **Tag Prediction (Multi-label):** For T possible tags, use a sigmoid output layer to predict tag probabilities $\hat{\mathbf{y}}_{tags} \in [0,1]^T$ from $\mathbf{z}_{final}$. The binary cross-entropy loss:

$$\mathcal{L}_{tags} = -\frac{1}{T}\sum_{t=1}^{T}\left(y_t \log \hat{y}_t + (1-y_t)\log(1-\hat{y}_t)\right)$$

  where $y_t \in \{0,1\}$ is the ground truth for tag t.

- **Category Prediction (Multi-class):** For C possible categories, use a softmax output layer to predict probabilities $\hat{\mathbf{y}}_{cat} \in [0,1]^C$. The cross-entropy loss:

$$\mathcal{L}_{cat} = -\sum_{c=1}^{C} y_c \log \hat{y}_c$$

  where $y_c \in \{0,1\}$ is the one-hot ground truth vector for the category.

Split	MSE (Reconstruction)	MAE (Reconstruction)
Train	0.018	0.091
Validation	0.022	0.104
Test	0.024	0.111

Table 1: Benchmarks for the autoencoder objective model

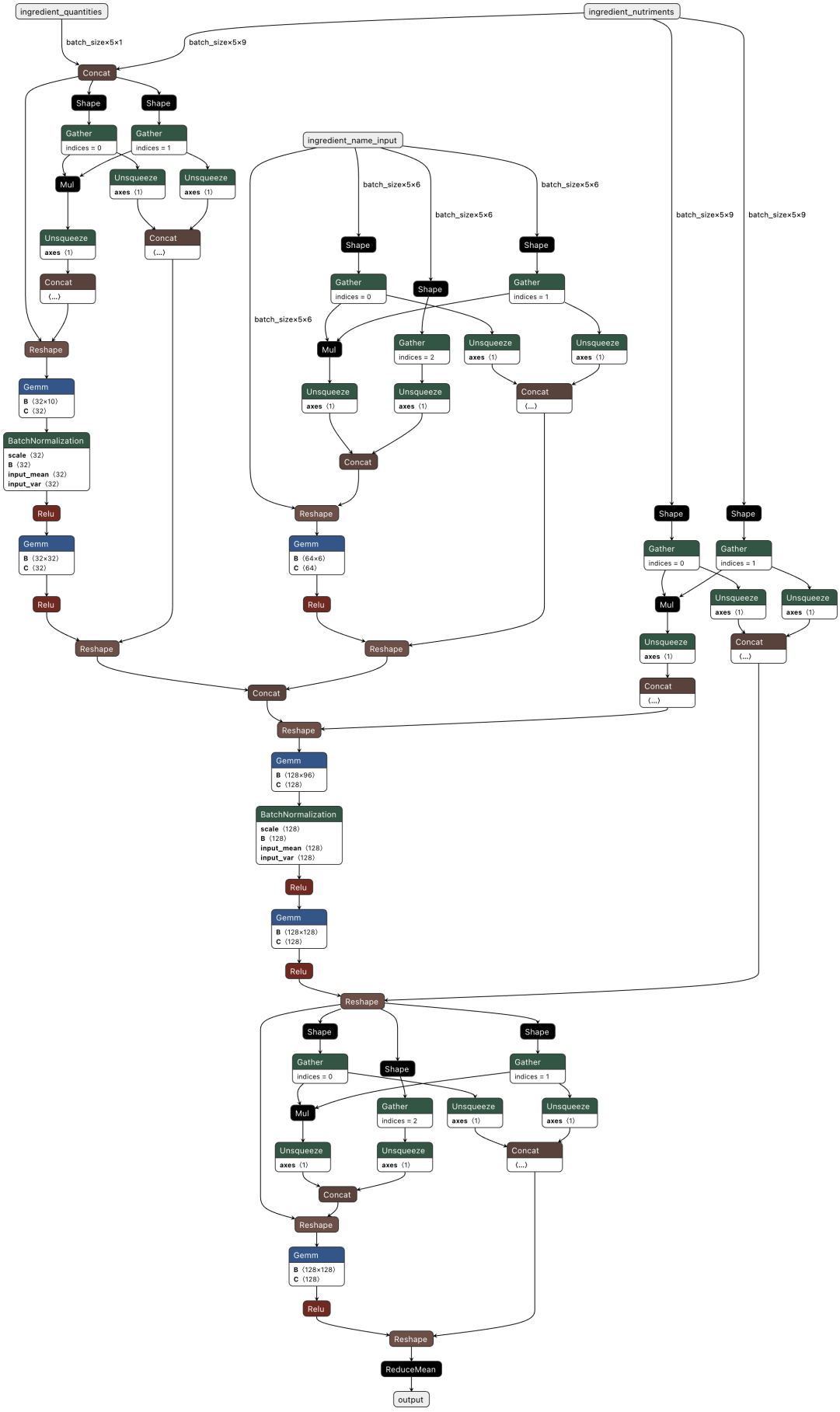
References

[BGM+20] Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland, December 2020. Association for Computational Linguistics.

[MLNM19] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating personalized recipes from historical user preferences. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5976–5982, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Wei] Alexander Wei. Food.com recipes with ingredients and tags. https://www.kaggle.com/datasets/realalexanderwei/food-com-recipes-with-ingredients-and-tags.

[MLNM19] [BGM+20] [Wei]


ingredient_quantities
ingredient_nutriments
batch_size×5×1
batch_size×5×9
Concat
Shape
Shape
Gather
indices = 0
Gather
indices = 1
Mul
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Concat
(...)
Concat
(...)
Reshape
Gemm
B (32×10)
C (32)
BatchNormalization
scale (32)
B (32)
input_mean (32)
input_var (32)
Relu
Gemm
B (32×32)
C (32)
Relu
Reshape
ingredient_name_input
batch_size×5×6
batch_size×5×6
batch_size×5×6
Shape
Shape
Gather
indices = 0
Shape
Gather
indices = 1
batch_size×5×6
Mul
Gather
indices = 2
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Concat
(...)
Concat
Reshape
Gemm
B (64×6)
C (64)
Relu
Reshape
batch_size×5×9
batch_size×5×9
Shape
Shape
Gather
indices = 0
Gather
indices = 1
Mul
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Concat
(...)
Concat
(...)
Concat
Reshape
Gemm
B (128×96)
C (128)
BatchNormalization
scale (128)
B (128)
input_mean (128)
input_var (128)
Relu
Gemm
B (128×128)
C (128)
Relu
Reshape
Shape
Shape
Gather
indices = 0
Shape
Gather
indices = 1
Mul
Gather
indices = 2
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Unsqueeze
axes (1)
Concat
(...)
Concat
Reshape
Gemm
B (128×128)
C (128)
Relu
Reshape
ReduceMean
output