# Electronic Supplementary Information

# A color-spectral machine learning path for five mixed amino acids analysis

Qiannan Duan, Jianchao Lee*, Shourong Zheng, Jiayuan Chen, Luo Run, Yunjin Feng, Zhaoyi Xu*

## Table of Contents

**Experiment section:**

**Figures and tables:**

**Experiment section**

## 1. Materials

All of chemical reagents (purity, ≥ 99%) were purchased from Aladdin Sigma-Aldrich, and were available used as received unless otherwise noted.

## 2. Setup of equipment

The core structure of the equipment is mainly composed of a light source **1**, a group of light equalizing chips **2**, a color-spectrum filter (named as CSF) **3**, a sample cell (a customized quartz cuvette with an optical path length of 10 mm, its maximum capacity of 100mL) **4** and a commercial HD camera (1080p/30fps, C920 PRO, Logitech) **5**. Among them, **1** (380-780 nm wavelength) is a combination of a 3-watt iodine tungsten lamp and a 0.25-watt white LED. **2** is prepared from some transparent slide of quartz glass (its thickness of 2 mm). And **3** is a color optical filter for modulating the light. In a dust-free operation, a designed color-printing template was printed on a transparent medical PET plate (in size of 50×50 mm) by a commercial inkjet printer (see Fig. S1), and then the printed plate was dried and sealed on the surface of a quartz slide.

## 3. Color-combined template design

A color-combined template (in size of 1200×1200 dpi, see in Fig. S1) was designed to guide the printing actions for preparing the CSF. It was consisted of an individual rectangle pattern included three overlapping C/M/Y color layers, referring to three inks: cyan, magenta and yellow. These layers were combined together by presetting the CMY values (0% to100% with 1% interval) in various directions.

## 4. Preparation of bulk solution

Two pretreated solutions were reserved as bulk solutions, including ultrapure water and human urine. Human urine solution was collected from three normal male volunteers (aged 22-25). The volunteers urinated directly into 250-mL glass beakers, and then the collected urine samples were mixed together in a 1000-mL beaker. Subsequently, the mixed sample was centrifuged at 12,000 rpm with 14,300 ×g for 15 min at 277 K to remove whole cells, large membrane fragments, and other debris. The supernatant was collected and filtered out the suspended particles using a PTFE micro-filtration membrane (pore size of 0.22 um), and placed in 277 K. Centrifuge (TGL-16M) was purchased from Lu Xiangyi Centrifuge Instrument Co., Ltd, Shanghai, China.

## 5. Mixed color agent preparation

A mixed color agent (MCA) contained following components that were dissolved in 400 mL of 0.2% (v/v) ethanol solution: 0.1g bromocresol green (BCG) sodium salt, 0.1g methyl red chloride, 0.1g bromothymol blue, 0.1g brilliant blue G250, and the pH was tuned to 7.8. This color stock solution was prepared ahead, and was diluted when it would be used. In every sampling operation, 2 mL MCA liquor was pipetted into the sample cell (Fig. 1B), and diluted with 48 mL bulk solution (ultrapure water or human urine solution).

## 6. Mixed amino acid samples preparation

All mixture samples were prepared from a series of AA stock solutions (histidine, arginine, phenylalanine, alanine threonine, proline and glycine), and were arranged to three systems (detail see Table S1), including (1) 5-dimensional AA (Hist-Arg-Phe-Alan-Thr) aqueous system, (2) 5-dimensional AA (Pro-Gly-His-Thr-Phe) aqueous system and (3) 5-dimensional AA (Pro-Gly-His-Thr-Phe) urine system. In these three designed systems, 5-dimensional AAs with various combinations were quantitatively added into the sample cell that pre-added 50mL mixed background solution (2 mL MCA+48 mL bulk solution), respectively. By using an automatic injection system, the AA fluids were controlled by five peristaltic pumps (model: BT300-2J YZ1515x, Linde Instrument co. LTD, Shanghai, China) equipped with the injection tubes. The rotating speed of the peristaltic pump is 3 rmp with a flow velocity of 1.95 mL/min by setting in advance. For one-batch operation, each amino acid was randomly distributed about between 0-1600 mg/L via controlling the injection time according to the requirements of the experiments. At the same time, there was a mini-blender for instantaneously mixing and uniformly agitating the sample with the chromogenic substrate. Here, the bulk solution could be selected from ultrapure water or human urine according to different researching needs.

## 7. Images acquisition and processing

All the samples were imaged as CSiMs when the light source kept stable. During the device running, the light beam (in wavelength of 380-780nm) **1** passed through **2**, **3** and **4** in turn. At the end of optical path, the camera **5** was used to detect the residual light, and the spectral property of sample was recorded as CSiM image. With the process of continuous injection, the changes of mixed solution were recorded in a 4-minute video, in which each time point corresponded to one mixture sample. For one-batch experiment, 1,200 images within one video were extracted by capturing one frame every 5 seconds. Ultimately, three mixed AA systems obtained 3,600, 3,600 and 6,000 CSiM images in a series of multiple-batch HTEs. Photo parameters are listed in Table S3. The image set collected from the mixed AA system I is exhibited in Fig. S4; the image sets of system II and III can be seen in Fig. S7. Correspondingly, their concentration values of all the samples were listed in Table S4-S6. And the photo collected from background substance (2 mL MCA+48 mL bulk solution, without any amino acids) on the same condition, was regarded as a blank control image.

All images came from different AA samples were uniformly cut to 300 × 300 pixels in size. Besides, all images from experimental groups were subtracted the blank image for uniformly correcting its background absorption interference. This step was implemented in IPython using spyder editor version 3.6.

## 8. Modeling

*Convolutional neural network algorithm*

Six CNN frameworks, LeNet-5, Vanilla CNN, RestNet-50, SqueezeNet, VGGNet and Inception v1 network, were selected. These frameworks (listed in Table S4) were used to extract local features directly from raw CSiM data, and explore non-linear relations between the CSiM and their concentration labels. For all the frameworks, their key functional block is a convolutional layer followed by nonlinear activation function.

$$\begin{cases} y_i^{'}=\sum_j w_{ij}x_j+b_i \\ y_i=\max(y_i^{'},0) \end{cases} \tag{1}$$

Where $x$ and $y$ are input and output vectors, respectively; and $w$ and $b$ are synaptic weights and biases, respectively. Where $y_i^{'}$ is an activation function (ReLU) for neurons in the hidden layers. The ReLU could avoid "gradient disappearing" during the computing processes. The principle of training a traditional network is to continuously optimize the network in $w$ (and $b$), which means a lot of training time in learning parameters. However, CNN can reduce the parameters in the training process through sharing weight and shrinking convolution kernels. The error-back-propagation algorithm can match the prediction value and factual value to the closest degree.

*Deep learning*

Google's Inception v1 CNN (a deep learning architecture with top-five accuracy in 2014 ImageNet Challenge) was imported as a primary function framework during the training process. We removed the final classification layer from the network and added a linear layer having 5 feature vectors, and retrained it with our dataset, fine-tuning the parameters across all layers. As shown in Table S3, the major layers of the network included multiple convolutions, ReLU nonlinearity, max pooling, fully connected layers, etc. And the framework is capable to avoid over-fitting problem by introducing dropout layers (reducing the dimension via eliminating the relevant parameters) and regularization layers.

During training an 80/20 split of training and test data were adopted, and each image was resized to 224×224 pixels for making it compatible with the original dimensions of the Inception v1. All layers of the network are fine-tuned using the learning rate of 0.001, max training grounds of 500 and batch size of 20, and the modified CNN was trained using backpropagation. In details, when a CSiM was taken as the input, the network processes the images through multiple hidden layers in the CNN model, and then outputs the predicting outcomes by a fully connected layer. Similarly, the same operation could be adopted for other five networks by merely replacing main network structures.

*Model Evaluation*

The statistical quality of the trained models was evaluated by two evaluation metrics: coefficient of determination ($R^2$), root mean squared error (RMSE). They are defined as following:

$$R^2 = [\frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}}]^2 \tag{2}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N}} \tag{3}$$

Where $x_i$ is the actual value, $y_i$ is the predicted value; $\bar{x}$, $\bar{y}$ are their means and $N$ is the number of samples.

## 9. Image similarity comparison

Four pairs of different parallel samples were selected randomly for testing the repeatability of CSiMs by image similarity comparison. Two trials, one was HASH algorithms and the other was a method of gray histograms.

HASH algorithms for estimating the image similarity was a process of examining the contents of an image by constructing a small hash code (or 'image fingerprint', a binary strings) that uniquely identified an image based on these contents. In this process, average hash (aHash), difference hash (dHash) and perceptual hash (pHash) algorithms were used to work on the difference between adjacent pixels, and Hamming distance (the number of bit positions in which the two bits are different while comparing two hash codes of equal length) was defined as the distance between parallel CSiMs. All the images were scaled down to 8×8 pixels to generate a 64 bit hash, and the image similarity was assessed by calculating Hamming distance. The similarity is defined as following:

$$Similarity = \frac{1 - H_{dist} \times 1.0}{m} \tag{4}$$

Where $m$ is 64 bit hash; $H_{dist}$ is the Hamming distance, measured by the aHash, dHash and pHash algorithm. The mean similarities on these three HASH algorithms were marked in Fig. S3A.

Gray histograms also were built based on four pairs of parallel CSiMs (see in Fig. S3B). The histogram, referring to a histogram of the pixel intensity values, is a graph showing the number of pixels in an image at each different gray intensity value found in that image. There were 256 different possible intensities (range from 0 to 255) for a grayscale image, so the histogram could graphically display 256 numbers showing the distribution of pixels amongst those grayscale values.

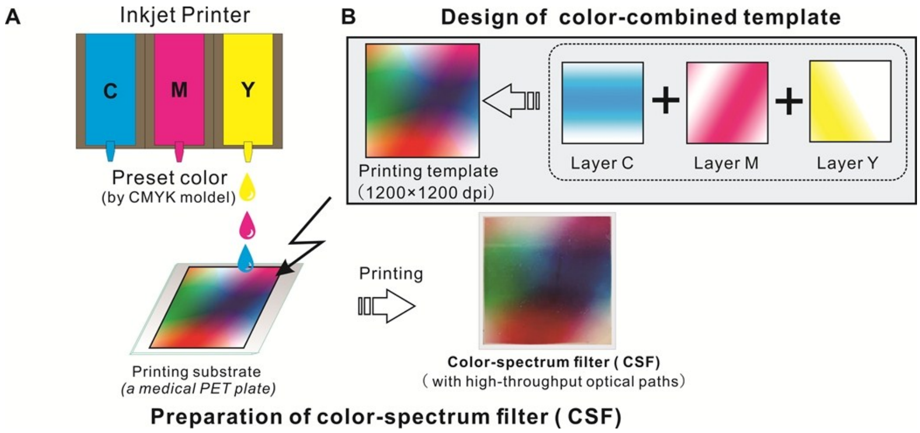

**Fig. S1** (A) CSF was manufactured by ink-jet printing technology. (B) Design of color-combined template.

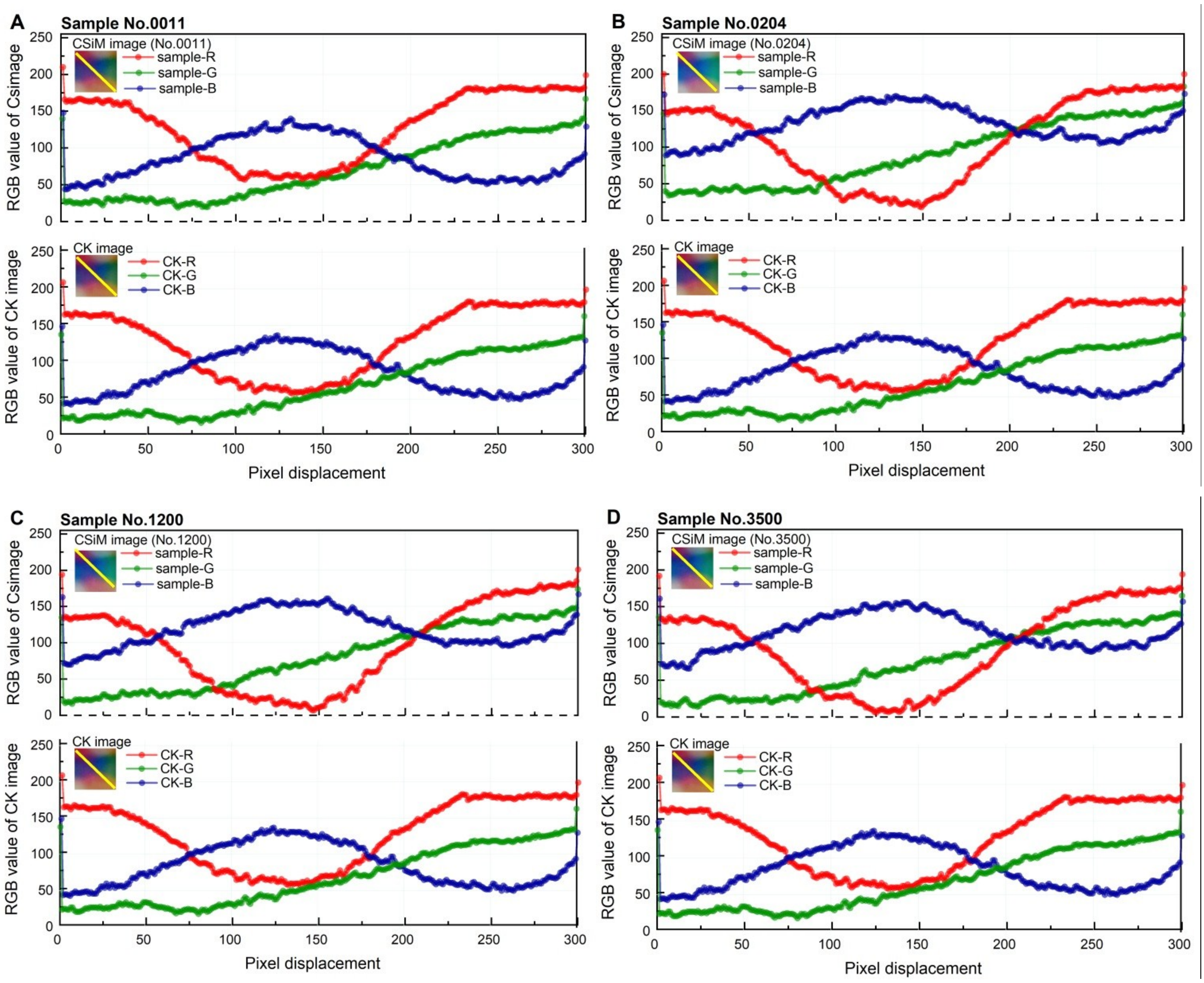

**Fig. S2** Variance analysis of the CSiM images between experimental and the CK groups. Spectral properties of samples could be observed from the differences of RGB values on the CSiMs, and these differences would be sensitively captured by the CNNs. The CK image was collected form the 50mL background solution (only containing the mixed color system without any AAs).

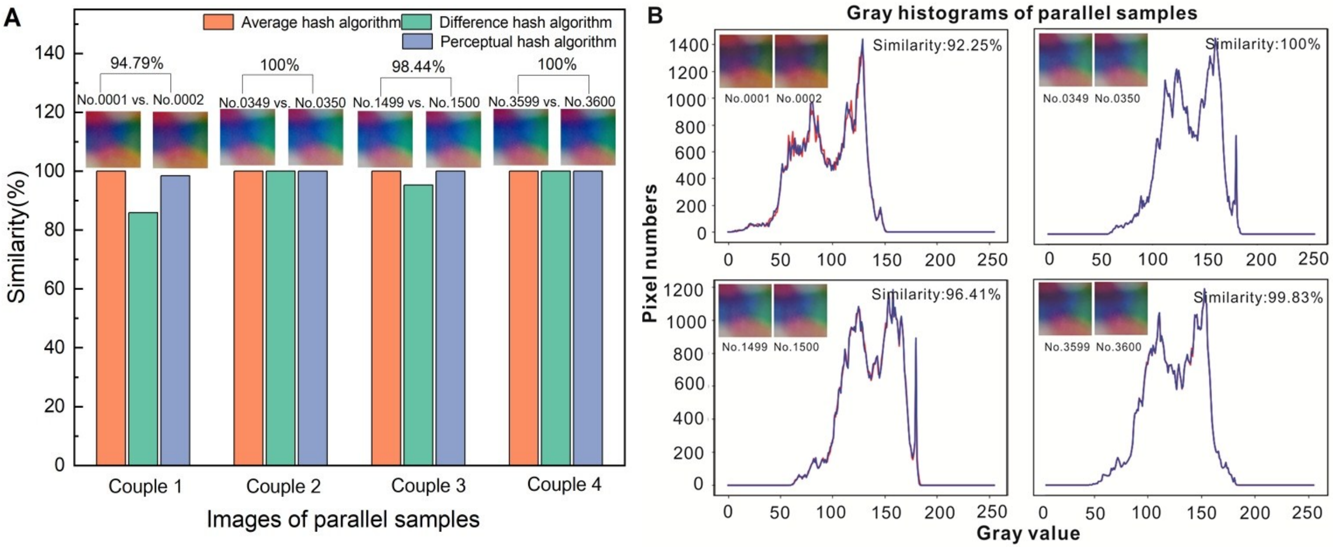

**Fig. S3** Image similarity comparison. (A) Three HASH algorithms (average hash algorithm, difference hash algorithm and perceptual hash algorithm), and (B) gray histograms were used to measure the similarity of the parallel CSiMs as well. Four pairs of different parallel CSiM images were selected randomly, and their mean similarities on three HASH algorithms were marked in (A).

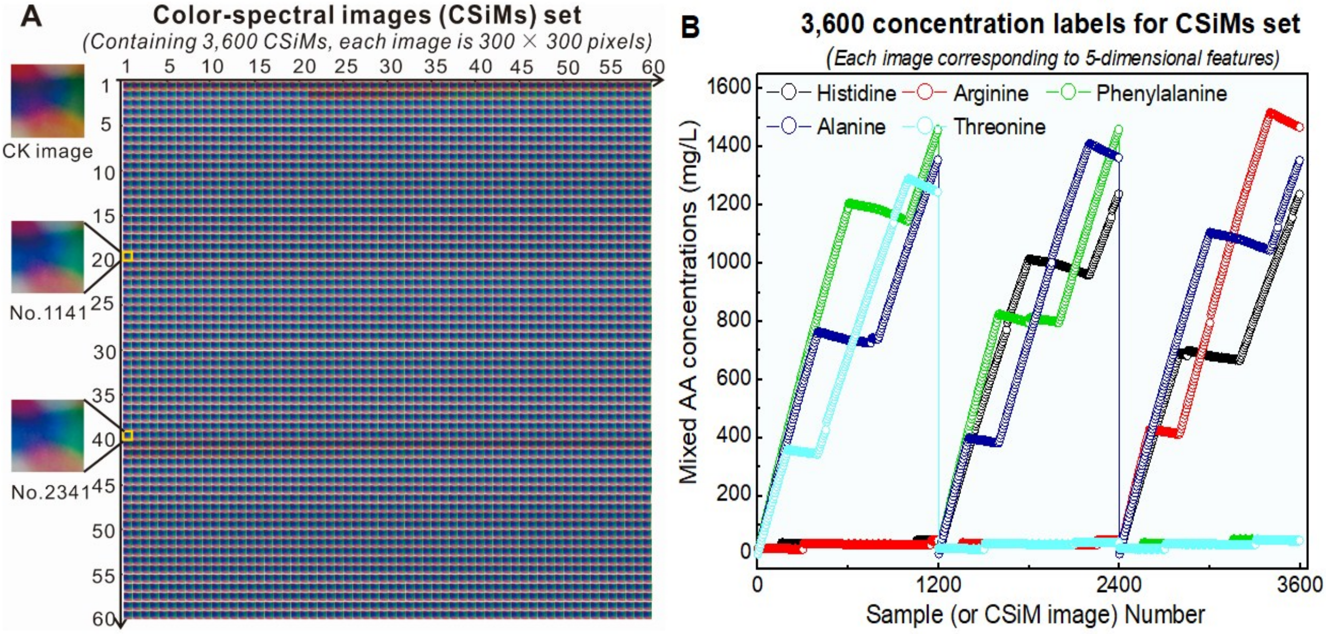

**Fig. S4** Obtained (A) CSiM images and (B) their 5-dimensional concentration labels of mixed amino acids.

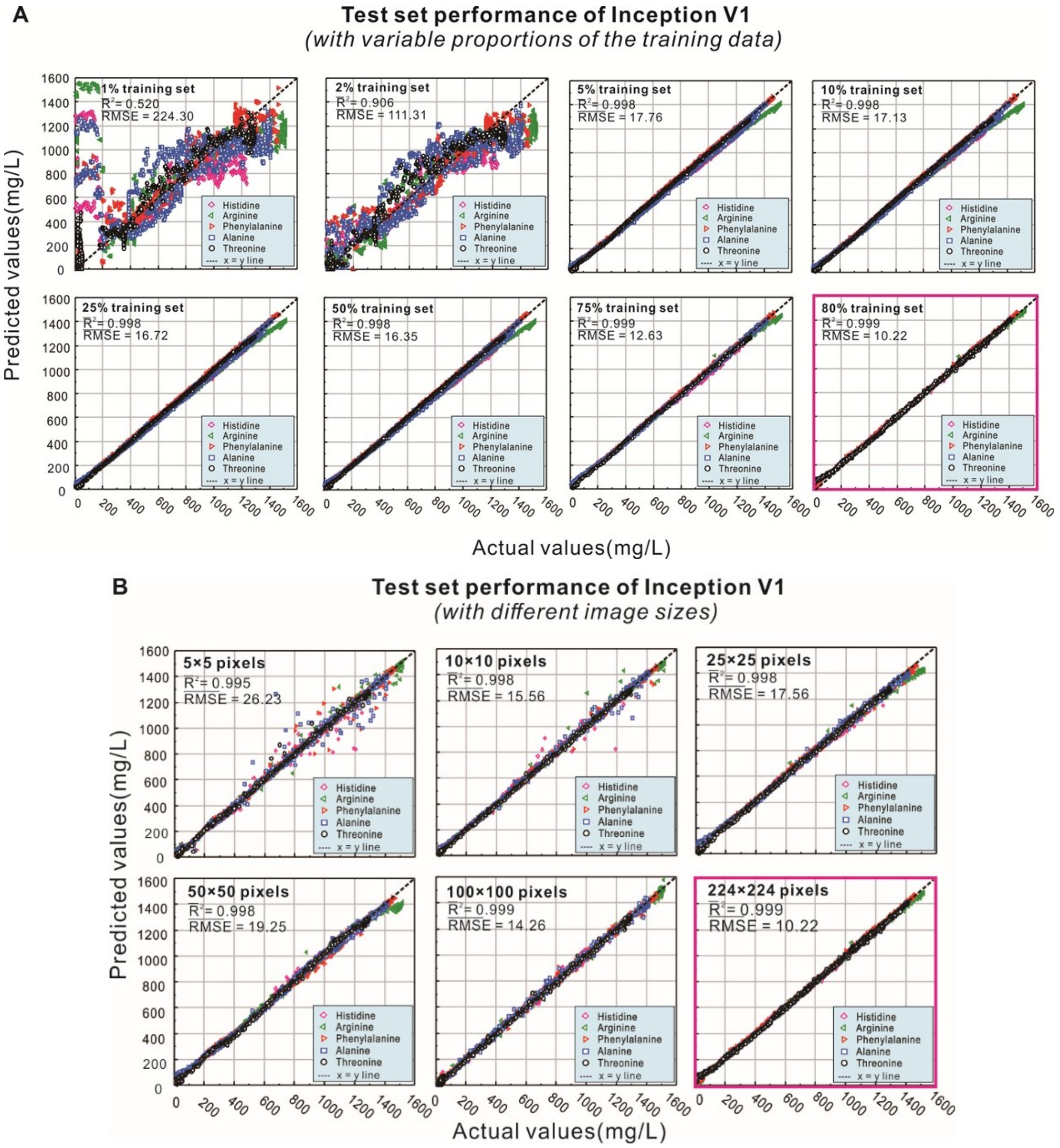

**Fig. S5** Test set performances of the Inception V1 model (A) at variable proportions of the training data and (B) different image sizes.

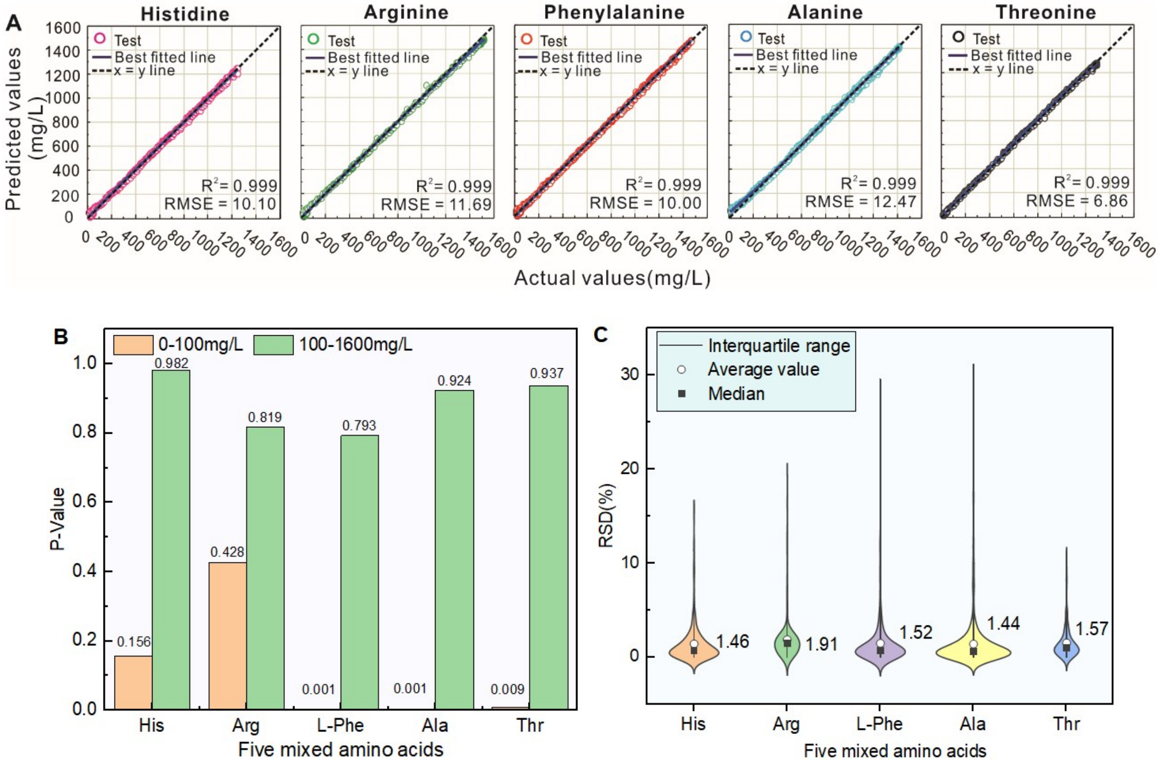

**Fig. S6** (A) Actual vs. predicted concentration plots for Inception v1 model. $R^2$ and RMSE of five AAs (His, Arg, L-phe, Ala and Thr) are used for measuring the predictive ability of the model, respectively, and only the test set shown in plots. (B) Difference significant of predicted vs. actual values on different AA concentration ranges. (C) Distribution of relative standard deviation (RSD) between predicted vs actual values. Violin plots of the RSD on various AAs, and only the concentration ranges from 100-1600mg/L shown. Mean RSDs of the sample are remarked.

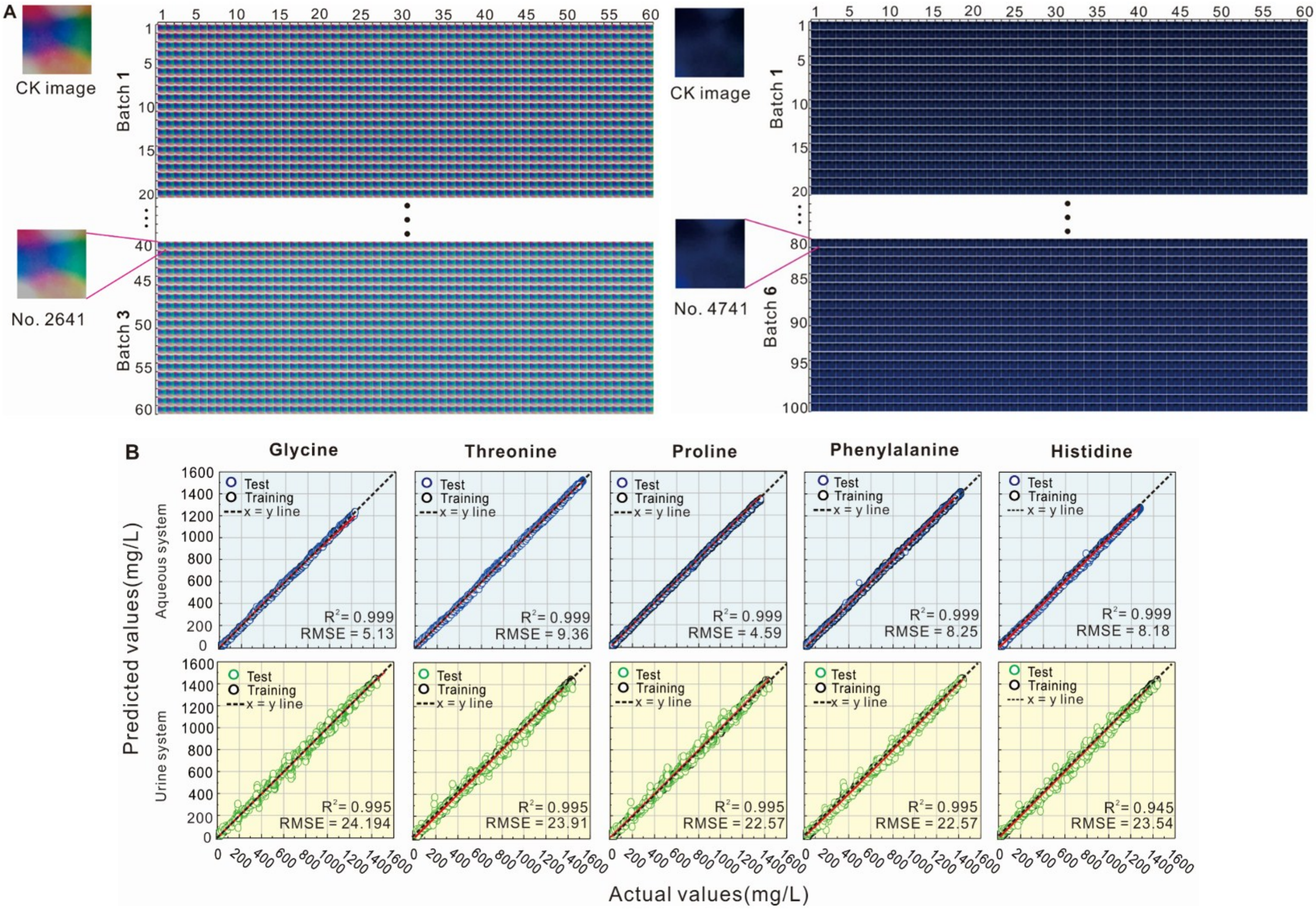

**Fig S7** (A) Image sets collected from 3,600 samples (containing 5 AAs: Pro, Gly, His, Thr and L-Phe) and 6,000 urine samples (containing 5 AAs: Pro, Gly, His, Thr and L-Phe). (B) Actual vs. predicted concentration plots for Inception v1 model. When the image sets being input, the linear fitting results between the outputs on the test set were displayed in plots. There are two AA mixed systems, including Pro-Gly-His-Thr-Ph aqueous system and Pro-Gly-His-Thr-Ph human urine system. $R^2$ and RMSE are used for measuring predictive ability of the model; dashed line represents y = x line; solid line represents best-fit line.

**Table S1 List of three mixed AA systems**

| Entry | Description | pH | Working Conc.<br>(g/L) | Temperature<br>(℃) |
|---|---|---|---|---|
| System I: 5-dimensional amino acid samples (bulk solution: water) | | | | |
| **1** | Histidine (His) | 8 | 20 | 25 |
| **2** | Arginine (Arg) | 8 | 20 | 25 |
| **3** | Phenylalanine (L-Phe) | 8 | 20 | 25 |
| **4** | Alanine (Ala) | 8 | 20 | 25 |
| **5** | Threonine (Thr) | 8 | 20 | 25 |
| System II: 5-dimensional amino acid samples (bulk solution: water) | | | | |
| **1** | Proline (Pro) | 8 | 20 | 25 |
| **2** | Glycine (Gly) | 8 | 20 | 25 |
| **3** | Histidine (His) | 8 | 20 | 25 |
| **4** | Threonine (Thr) | 8 | 20 | 25 |
| **5** | Phenylalanine (L-Phe) | 8 | 20 | 25 |
| System III: 5-dimensional amino acid samples (bulk solution: human urine) | | | | |
| **1** | Proline (Pro) | 8 | 20 | 25 |
| **2** | Glycine (Gly) | 8 | 20 | 25 |
| **3** | Histidine (His) | 8 | 20 | 25 |
| **4** | Threonine (Thr) | 8 | 20 | 25 |
| **5** | Phenylalanine (L-Phe) | 8 | 20 | 25 |

**Table S2 Photo Parameters**

| Image resolution | Bit depth | Horizontal resolution | Vertical resolution | Color model |
|---|---|---|---|---|
| 300×300 | 24 | 300 dpi | 300dpi | RGB |

**Table S3 Detailed architecture of CNNs**

| Google Inception v1 CNN | | |
|---|---|---|
| **Layer** | **Layer(type)** | **Output Shape** |
| 0 | Input Layer | (None, 224,224,3) |
| 1 | Convolution Layer | (None, 112,112,64) |
| 2 | Relu function | (None, 112,112,64) |
| 3 | Padding Layer | (None, 113, 113, 64) |
| 4 | Max Pooling Layer | (None, 56, 56, 64) |
| 5 | Local Response Normalization Layer | (None, 56, 56, 64) |
| 6 | Convolution Layer | (None, 56, 56, 64) |
| 7 | Relu function | (None, 56, 56, 64) |
| 8 | Convolution Layer | (None, 56, 56, 192) |
| 9 | Relu function | (None, 56, 56, 192) |
| 10 | Local Response Normalization Layer | (None, 56, 56, 192) |
| 11 | Max Padding Layer | (None, 57, 57, 192) |
| 12 | Pooling Layer | (None, 28, 28, 192) |
| 13 | Inception_Net Graph | (None, 28, 28, 256) |

|  |  |  |
|---|---|---|
| 14 | Inception_Net Graph | (None, 28, 28, 480) |
| 15 | Padding Layer | (None, 29, 29, 480) |
| 16 | Max Pooling Layer | (None, 14, 14, 480) |
| 17 | Inception_Net Graph | (None, 14, 14, 512) |
| 18 | Inception_Net Graph | (None, 14, 14, 512) |
| 19 | Inception_Net Graph | (None, 14, 14, 512) |
| 20 | Inception_Net Graph | (None, 14, 14, 528) |
| 21 | Inception_Net Graph | (None, 14, 14, 832) |
| 22 | Padding Layer | (None, 15, 15, 832) |
| 23 | Max Pooling Layer | (None, 7, 7, 832) |
| 24 | Inception_Net Graph | (None, 7, 7, 832) |
| 25 | Inception_Net Graph | (None, 7, 7, 1024) |
| 26 | Max Pooling Layer | (None, 1, 1, 1024) |
| 27 | Dropout Layer | (None, 1, 1, 1024) |
| 28 | Flatten Layer | (None, 1024) |
| 29 | Linear Layer | (None, 1000) |
| 30 | Ramp | (None, 1000) |
| 31 | Linear Layer | (None, 5) |
| 32 | Output | (None, 5) |

| **RestNet-50** |  |  |
|---|---|---|
| **Layer** | **Layer(type)** | **Output Shape** |
| 0 | Input Layer | (None, 224, 224, 3) |
| 1 | Convolution Layer | (None, 112, 112, 64) |
| 2 | Batch Normalization Layer | (None, 112, 112, 64) |
| 3 | Ramp | (None, 112, 112, 64) |
| 4 | Padding Layer | (None, 113, 113, 64) |
| 5 | Pooling Layer | (None, 56, 56, 64) |
| 6 | NetGraph (12 nodes) | (None, 56, 56, 256) |
| 7 | NetGraph (10 nodes) | (None, 56, 56, 256) |
| 8 | NetGraph (10 nodes) | (None, 56, 56, 256) |
| 9 | NetGraph (12 nodes) | (None, 28, 28, 512) |
| 10 | NetGraph (10 nodes) | (None, 28, 28, 512) |
| 11 | NetGraph (10 nodes) | (None, 28, 28, 512) |
| 12 | NetGraph (10 nodes) | (None, 28, 28, 512) |
| 13 | NetGraph (12 nodes) | (None, 14, 14, 1024) |
| 14 | NetGraph (10 nodes) | (None, 14, 14, 1024) |
| 15 | NetGraph (10 nodes) | (None, 14, 14, 1024) |
| 16 | NetGraph (10 nodes) | (None, 14, 14, 1024) |
| 17 | NetGraph (10 nodes) | (None, 14, 14, 1024) |
| 18 | NetGraph (10 nodes) | (None, 14, 14, 1024) |
| 19 | NetGraph (12 nodes) | (None, 7, 7, 2048) |
| 20 | NetGraph (10 nodes) | (None, 7, 7, 2048) |
| 21 | NetGraph (10 nodes) | (None, 7, 7, 2048) |
| 22 | Pooling Layer | (None, 1, 1, 2048) |
| 23 | Flatten Layer | Vector (None, 2048) |

| | | |
|---|---|---|
| 24 | Linear Layer | Vector (None, 1000) |
| 25 | Ramp | Vector (None, 1000) |
| 26 | Linear Layer | Vector (None, 5) |
| 27 | Output | Vector (None, 5) |

**VGG-16**

| Layer | Layer(type) | Output Shape |
|---|---|---|
| 0 | Input Layer | (None, 224, 224, 3) |
| 1 | Convolution Layer | (None, 224, 224, 64) |
| 2 | Ramp | (None, 224, 224, 64) |
| 3 | Convolution Layer | (None, 224, 224, 64) |
| 4 | Ramp | (None, 224, 224, 64) |
| 5 | Pooling Layer | (None, 112, 112, 64) |
| 6 | Convolution Layer | (None, 112, 112, 128) |
| 7 | Ramp | (None, 112, 112, 128) |
| 8 | Convolution Layer | (None, 112, 112, 128) |
| 9 | Ramp | (None, 112, 112, 128) |
| 10 | Pooling Layer | (None, 56, 56, 128) |
| 11 | Convolution Layer | (None, 56, 56, 256) |
| 12 | Ramp | (None, 56, 56, 256) |
| 13 | Convolution Layer | (None, 56, 56, 256) |
| 14 | Ramp | (None, 56, 56, 256) |
| 15 | Convolution Layer | (None, 56, 56, 256) |
| 16 | Ramp | (None, 56, 56, 256) |
| 17 | Pooling Layer | (None, 28, 28, 512) |
| 18 | Convolution Layer | (None, 28, 28, 512) |
| 19 | Ramp | (None, 28, 28, 512) |
| 20 | Convolution Layer | (None, 28, 28, 512) |
| 21 | Ramp | (None, 28, 28, 512) |
| 22 | Convolution Layer | (None, 28, 28, 512) |
| 23 | Ramp | (None, 28, 28, 512) |
| 24 | Pooling Layer | (None, 14, 14, 512) |
| 25 | Convolution Layer | (None, 14, 14, 512) |
| 26 | Ramp | (None, 14, 14, 512) |
| 27 | Convolution Layer | (None, 14, 14, 512) |
| 28 | Ramp | (None, 14, 14, 512) |
| 29 | Convolution Layer | (None, 14, 14, 512) |
| 30 | Ramp | (None, 14, 14, 512) |
| 31 | Pooling Layer | (None, 7, 7, 512) |
| 32 | Flatten Layer | (None, 25088) |
| 33 | Linear Layer | (None, 4096) |
| 34 | Ramp | (None, 4096) |
| 35 | Dropout Layer | (None, 4096) |
| 36 | Linear Layer | (None, 4096) |
| 37 | Ramp | (None, 4096) |
| 38 | Dropout Layer | (None, 4096) |

| | | |
|---|---|---|
| 39 | Linear Layer | (None, 101) |
| 40 | Ramp | (None, 101) |
| 41 | Linear Layer | (None, 5) |
| 42 | Output | (None, 5) |

**SqueezeNet V1.1**

| Layer | Layer(type) | Output Shape |
|---|---|---|
| 0 | Input Layer | (None, 227, 227, 3) |
| 1 | Convolution Layer | (None, 113, 113, 64) |
| 2 | Ramp | (None, 113, 113, 64) |
| 3 | Pooling Layer | (None, 56, 56, 64) |
| 4 | NetGraph (7 nodes) | (None, 56, 56, 128) |
| 5 | NetGraph (7 nodes) | (None, 56, 56, 128) |
| 6 | Padding Layer | (None, 57, 57, 128) |
| 7 | Pooling Layer | (None, 28, 28, 128) |
| 8 | NetGraph (7 nodes) | (None, 28, 28, 256) |
| 9 | NetGraph (7 nodes) | (None, 28, 28, 256) |
| 10 | Padding Layer | (None, 29, 29, 256) |
| 11 | Pooling Layer | (None, 14, 14, 256) |
| 12 | NetGraph (7 nodes) | (None, 14, 14, 384) |
| 13 | NetGraph (7 nodes) | (None, 14, 14, 384) |
| 14 | NetGraph (7 nodes) | (None, 14, 14, 512) |
| 15 | NetGraph (7 nodes) | (None, 14, 14, 512) |
| 16 | Dropout Layer | (None, 14, 14, 512) |
| 17 | Convolution Layer | (None, 14, 14, 1000) |
| 18 | Ramp | (None, 14, 14, 1000) |
| 19 | Aggregation Layer | (None, 1000) |
| 20 | Flatten Layer | (None, 1000) |
| 21 | Ramp | (None, 1000) |
| 22 | Linear Layer | (None, 5) |
| 23 | Output | (None, 5) |

**Vanllia CNN**

| | Layer(type) | Output Shape |
|---|---|---|
| 0 | Input Layer | (None, 40, 40, 3) |
| 1 | Convolution Layer | (None, 40, 40, 16) |
| 2 | Activation TangH1 Layer | (None, 40, 40, 16) |
| 3 | ActivationAbs1 Layer | (None, 40, 40, 16) |
| 4 | Pooling Layer | (None, 20, 20, 16) |
| 5 | Convolution Layer | (None, 20, 20, 48) |
| 6 | Activation TangH2 Layer | (None, 20, 20, 48) |
| 8 | ActivationAbs2 Layer | (None, 20, 20, 48) |
| 9 | Pooling Layer | (None, 10, 10, 48) |
| 10 | Convolution Layer | (None, 8, 8, 64) |
| 11 | Activation TangH3 Layer | (None, 8, 8, 64) |
| 12 | ActivationAbs3 Layer | (None, 8, 8, 64) |

| | | |
|---|---|---|
| 13 | Padding Layer | (None, 9, 9, 64) |
| 14 | Pooling Layer | (None, 4, 4, 64) |
| 15 | Convolution Layer | (None, 3, 3, 64) |
| 16 | Activation TangH4 Layer | (None, 3, 3, 64) |
| 17 | ActivationAbs4 Layer | (None, 3, 3, 64) |
| 18 | Flatten Layer | (None, 567) |
| 19 | Linear Layer | (None, 100) |
| 20 | Activation TangH5 Layer | (None, 100) |
| 21 | ActivationAbs5 Layer | (None, 100) |
| 22 | Linear Layer | (None, 10) |
| 23 | Reshape Layer | (None, 5, 2) |
| 24 | Ramp Layer | (None, 5, 2) |
| 25 | Linear Layer | (None, 5) |
| 26 | Output | (None, 5) |

**LeNet-5**

| Layer | Layer(type) | Output Shape |
|---|---|---|
| 0 | Input Layer | (None, 28, 28, 3) |
| 1 | Convolution Layer | (None, 40, 40, 20) |
| 2 | Ramp Layer | (None, 40, 40, 20) |
| 3 | Pooling Layer | (None, 12, 12, 20) |
| 4 | Convolution Layer | (None, 8, 8, 50) |
| 5 | Ramp Layer | (None, 8, 8, 50) |
| 6 | Pooling Layer | (None, 4, 4, 50) |
| 7 | Flatten Layer | (None, 800) |
| 8 | Linear Layer | (None, 500) |
| 9 | Ramp Layer | (None, 500) |
| 10 | Linear Layer | (None, 5) |
| 11 | Output | (None, 5) |