convnn_r_plots

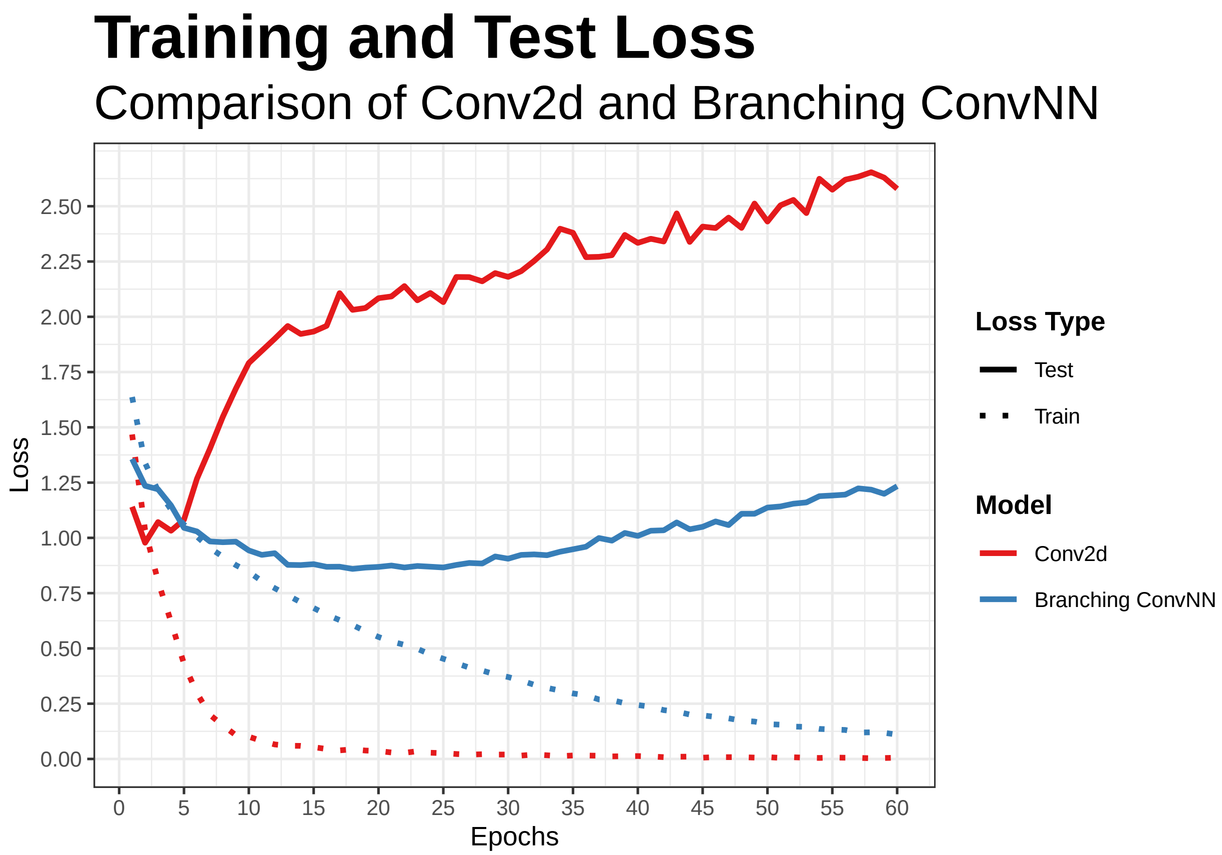
2025-09-27

```r
## Loss Plot
# "Convolutional-Nearest-Neighbor/Output/Sep_24_Branching_NoSplit/vgg_1e-5_cos/CIFAR10/Col_Col_Branch/"

setwd("/Users/mingikang/Developer/Convolutional-Nearest-Neighbor/plots")
data = read.csv("csv/Loss_Comparison.csv")

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Assuming 'data' is loaded correctly, we create the long format data frame
df_long <- data %>%
  pivot_longer(
    cols = -epoch,
    names_to = c("Model", "Type", ".value"),
    names_sep = "_"
  )

# Create a single combined plot
combined_plot <- df_long %>%
  ggplot(aes(x = epoch, y = Loss, color = Model, linetype = Type)) +
  geom_line(linewidth = 1.0) +

  # --- Manual Scales for Aesthetics ---
  scale_color_brewer(
    palette = "Set1",
    labels = c("ConvNN" = "Branching ConvNN")
    ) +

  scale_linetype_manual(
    name = "Loss Type", # Legend title for linetype
    values = c("Train" = "dotted", "Test" = "solid") # Assign specific linetypes
  ) +

  scale_x_continuous(
```

```r
    breaks = seq(0, 60, by=5)
  ) +
  scale_y_continuous(
    breaks = seq(0, 2.5, by=0.25)
  ) +

  # --- Labels and Titles ---
  labs(
    title = "Training and Test Loss",
    subtitle = "Comparison of Conv2d and Branching ConvNN",
    x = "Epochs",
    y = "Loss",
    color = "Model" # Legend title for color
  ) +

  # --- Theme and Styling ---
  theme_bw(base_size = 10) +
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold", size = 23),
    plot.subtitle = element_text(size = 18),
    legend.title = element_text(face = "bold") # Make legend titles bold
  )

# Display the combined plot
print(combined_plot)
```




```r
# Save the combined plot to a file
ggsave(
  "csv/loss_comparison_plot.png",
  plot = combined_plot,
  width = 8,
  height = 5,
  units = "in",
  dpi = 300,
  bg = "white"
)
```

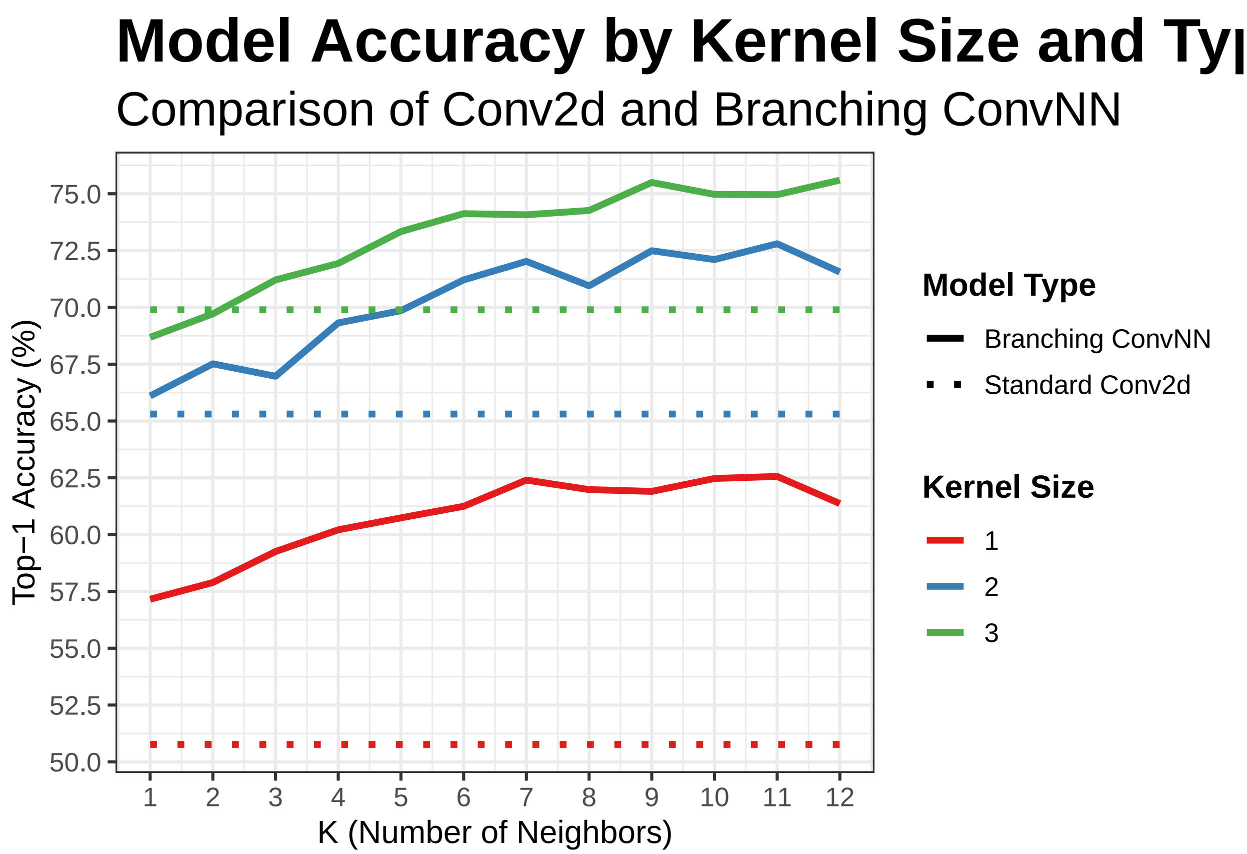
```r
## Ks Plot # Load the necessary libraries
# "Output/Sep_25_Branching_NoSplit_KTest/vgg_1e-5_cos/CIFAR10/LocCol_LocCol_Branch"


library(tidyverse)
# The stringr package (part of tidyverse) is used for str_detect()
library(stringr)

# --- Data Reading ---
# Set your working directory and read the data using the robust read_csv()
setwd("/Users/mingikang/Developer/Convolutional-Nearest-Neighbor/plots")
df <- read_csv("csv/Ks_Comparison.csv")
```

```
## Rows: 12 Columns: 7
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): K, Ks1_K, Ks2_K, Ks3_K, Ks1, Ks2, Ks3
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# --- Step 1: Reshape and Process Data ---
# Reshape the data and create new columns for Kernel Size and Model Type
df_processed <- df %>%
  pivot_longer(
    cols = -K,
    names_to = "Metric",
    values_to = "Value"
  ) %>%
  mutate(
    # Create a 'Kernel_Size' column by extracting the number from the 'Metric' string
    Kernel_Size = case_when(
      str_detect(Metric, "Ks1") ~ "1",
      str_detect(Metric, "Ks2") ~ "2",
      str_detect(Metric, "Ks3") ~ "3"
    ),
    # Create a 'Model_Type' column based on whether '_K' is in the 'Metric' string
    Model_Type = case_when(
      str_detect(Metric, "_K") ~ "Branching ConvNN",
      TRUE                     ~ "Standard Conv2d" # 'TRUE' acts as an else condition
    )
  )

# --- Step 2: Create the Plot with New Aesthetics ---
# Map 'color' to Kernel_Size and 'linetype' to Model_Type
k_plot <- ggplot(df_processed, aes(x = K, y = Value, color = Kernel_Size, linetype = Model_Type)) +
  geom_line(linewidth = 1.2) + # Draw the lines

  # --- Customize Scales and Legends ---
  scale_color_brewer(
    palette = "Set1",
    name = "Kernel Size" # Sets the title for the color legend
  ) +
  scale_linetype_manual(
    name = "Model Type", # Sets the title for the linetype legend
    values = c("Standard Conv2d" = "dotted", "Branching ConvNN" = "solid")
  ) +

  # --- Customize Axes and Labels ---
  scale_x_continuous(breaks = 1:12) + # Ensure integer ticks for K
  scale_y_continuous(breaks = seq(50, 77.25, 2.5)) + # Ensure integer ticks for K

  labs(
    title = "Model Accuracy by Kernel Size and Type",
    subtitle = "Comparison of Conv2d and Branching ConvNN",
    x = "K (Number of Neighbors)",
    y = "Top-1 Accuracy (%)"
  ) +
```

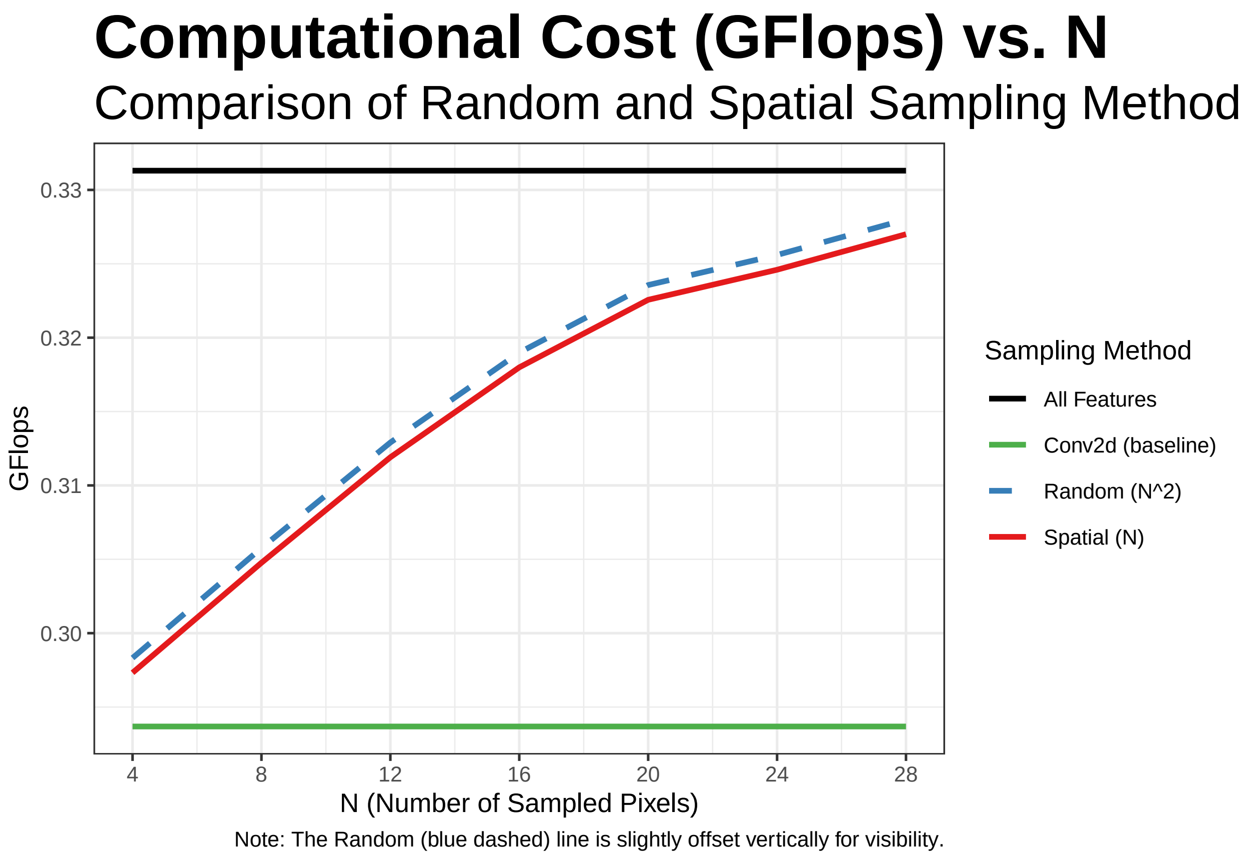
```r
  # --- Apply a Theme ---
  theme_bw(base_size = 12) +
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold", size = 23),
    plot.subtitle = element_text(size=18),
    legend.title = element_text(face = "bold") # Make legend titles bold
  )

# Step 3: Display the plot
print(k_plot)
```



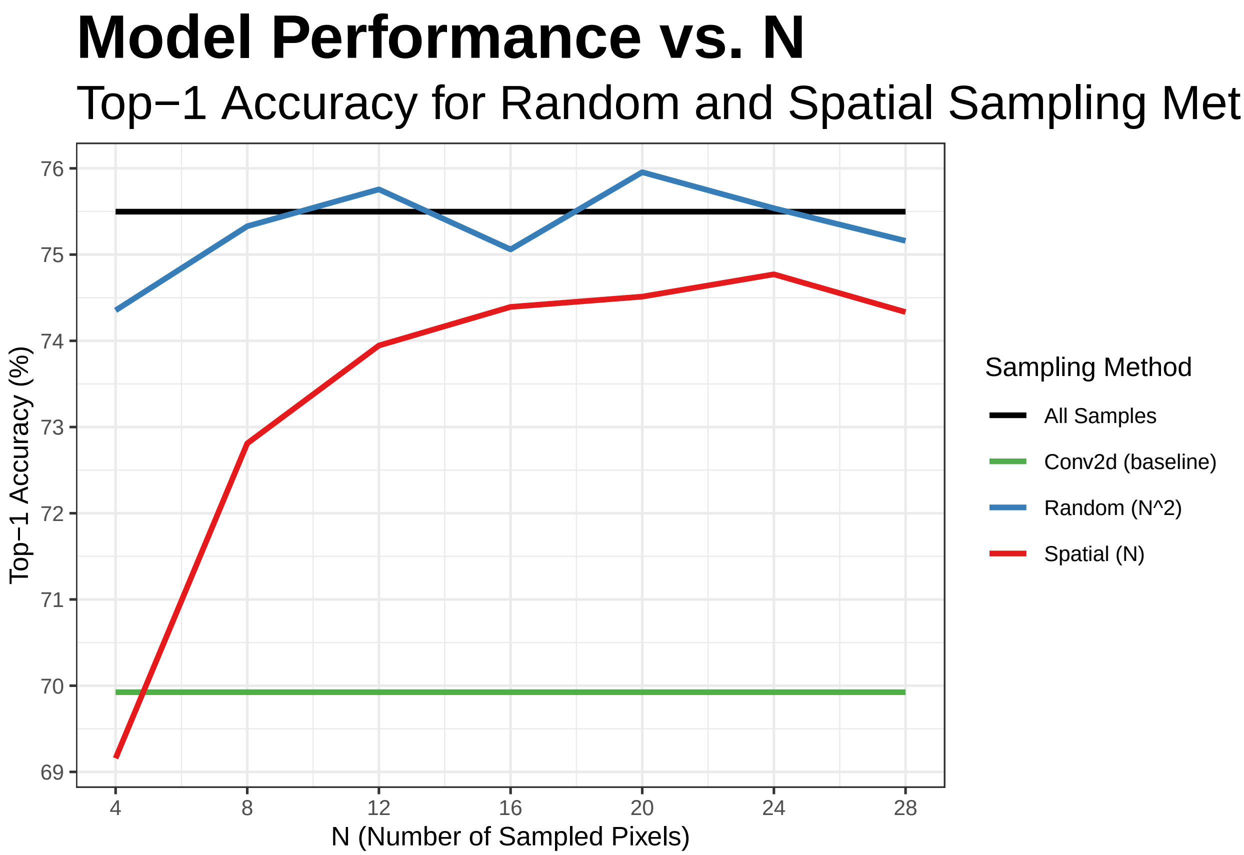

```r
# Step 4: Save the plot to a file
ggsave(
  "csv/ks_comparison_plot.png",
  plot = k_plot,
  width = 10,
  height = 6,
  units = "in",
  dpi = 300,
  bg = "white"
)
```

```r
## Random + Spatial Sampling Plot
# "Convolutional-Nearest-Neighbor/Output/Sep_27_Branching_NTest/vgg_1e-5_cos/CIFAR10/LocCol_LocCol_Bran
# Make sure the necessary library is loaded
library(tidyverse)

# --- Step 1: Load Data (Your code) ---
setwd("/Users/mingikang/Developer/Convolutional-Nearest-Neighbor/plots")
df = read.csv("csv/N_Samples_Comparison-oct6.csv")

# --- Step 2: Reshape the Data for Plotting (Your code) ---
df_long <- df %>%
  select(N, Rand_GFlops, Rand_Top1, Spat_GFlops, Spat_Top1, All_GFlops, All_Top1, Conv_GFlops, Conv_Top
  pivot_longer(
    cols = -N,
    names_to = c("Type", ".value"),
    names_sep = "_"
  )

# --- NEW: Define an offset value and apply it to the data ---
# This creates a new data frame with a new column for the offset GFlops values.
offset_value <- 0.001
df_long_offset <- df_long %>%
  mutate(GFlops_offset = GFlops + if_else(Type == "Rand", offset_value, 0.0))

# --- Step 3: Create the GFlops Plot with the offset data ---
gflops_plot <- ggplot(df_long_offset, aes(x = N, y = GFlops_offset, color = Type, linetype = Type)) + #
  geom_line(linewidth = 1) +

  # --- Color scale (Your code) ---
  scale_color_manual(
    name = "Sampling Method",
    labels = c(Rand = "Random (N^2)", Spat = "Spatial (N)", All = "All Features", Conv = "Conv2d (basel
    values = c(Rand = "#377EB8", Spat = "#E41A1C", All = "black", Conv = "#4DAF4A")
  ) +

  # --- NEW: Add a manual linetype scale to make the "Random" line dashed ---
  scale_linetype_manual(
    name = "Sampling Method",
    labels = c(Rand = "Random (N^2)", Spat = "Spatial (N)", All = "All Features", Conv = "Conv2d (basel
    values = c(Rand = "dashed", Spat = "solid", All = "solid", Conv = "solid")
  ) +

  # --- Customize Axes and Labels ---
  scale_x_continuous(breaks = seq(4, 28, 4)) +
  scale_y_continuous(breaks = seq(0.27, 0.34, 0.01)) +

  # --- NEW: Update labs to remove the redundant color title and add a caption ---
  labs(
    title = "Computational Cost (GFlops) vs. N",
    subtitle = "Comparison of Random and Spatial Sampling Methods",
    x = "N (Number of Sampled Pixels)",
    y = "GFlops",
    caption = "Note: The Random (blue dashed) line is slightly offset vertically for visibility."
```

```r
  ) +
  theme_bw(base_size = 10) +
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold", size = 23),
    plot.subtitle = element_text(size = 18)
  )

# --- Display the final plot ---

# --- Step 4: Create the Top-1 Accuracy Plot ---
top1_plot <- ggplot(df_long, aes(x = N, y = Top1, color = Type)) +
  geom_line(linewidth = 1) +
    # --- NEW: Use the same manual color scale ---
  scale_color_manual(
    name = "Sampling Method", # Legend title
    labels = c(
      Rand = "Random (N^2)",
      Spat = "Spatial (N)",
      All = "All Samples",
      Conv = "Conv2d (baseline)"
    ),
    values = c(
      Rand = "#377EB8",  # Blue
      Spat = "#E41A1C",  # Red
      All = "black",     # Black
      Conv = "#4DAF4A"   # Green
    )
  ) +


  # --- Customize Axes and Labels ---
  scale_x_continuous(breaks = seq(4, 28, 4)) + # Ensure integer ticks for N
  scale_y_continuous(breaks = seq(67, 76, 1)) + # Ensure integer ticks for K

  labs(
    title = "Model Performance vs. N",
    subtitle = "Top-1 Accuracy for Random and Spatial Sampling Methods",
    x = "N (Number of Sampled Pixels)",
    y = "Top-1 Accuracy (%)",
    color = "Sampling Method"
  ) +
  theme_bw(base_size = 10) +
  theme(legend.position = "right",
        plot.title = element_text(face = "bold", size = 23),
        plot.subtitle = element_text(size = 18)
        )


# --- Step 5: Display the Plots ---
print(gflops_plot)
```




```r
print(top1_plot)
```




```
# --- Step 6: (Optional) Save the Plots to Files ---
ggsave("csv/N_Gflops_New.png", plot = gflops_plot, width = 8, height = 5, dpi = 300, bg = "white")
ggsave("csv/N_Accuracy_New.png", plot = top1_plot, width = 8, height = 5, dpi = 300, bg = "white")
```